THESIS

USE OF AN ACOUSTIC NETWORK AS AN UNDERWATER POSITIONING SYSTEM

by

Michael S. Reed

June 2006

Thesis Advisors:	Joseph A. Rice Roberto Cristi

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Use of an Acoustic Network as an Underwater Positioning System			**5. FUNDING NUMBERS**
6. AUTHOR(S) Michael S. Reed			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT **Approved for public release; distribution is unlimited**			**12b. DISTRIBUTION CODE** A
13. ABSTRACT (maximum 200 words) Underwater acoustic networks provide an interface between UUVs and surface or land-based control systems. By exploiting range data measured incidental to communications on these networks it is possible to perform underwater positioning similar to that of the satellite-based GPS program. In this thesis, several algorithms for generating position fixes from these range data are implemented, tested, and evaluated with synthetic data. The algorithms are then applied to data obtained during operations at sea.			
14. SUBJECT TERMS acoustic, ranging, Seaweb, undersea warfare, navigation, submarine, UUV, AUV			**15. NUMBER OF PAGES** 87
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

USE OF AN ACOUSTIC NETWORK AS AN UNDERWATER POSITIONING SYSTEM

Michael S. Reed
Lieutenant, United States Navy
B.S., Case Western Reserve University, 1999

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2006

Author:	Michael S. Reed
Approved by:	Joseph A. Rice Thesis Advisor
	Roberto Cristi Co-Advisor
	Jeffrey B. Knorr Chairman, Department of Electrical and Computer Engineering
	Donald P. Brutzman Chair, Undersea Warfare Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Underwater acoustic networks provide an interface between UUVs and surface or land-based control systems. By exploiting range data measured incidental to communications on these networks it is possible to perform underwater positioning similar to that of the satellite-based GPS program. In this thesis, several algorithms for generating position fixes from these range data are implemented, tested, and evaluated with synthetic data. The algorithms are then applied to data obtained during operations at sea.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First I would like to thank my advisors, Prof. Joe Rice and Dr. Roberto Cristi. Their expert knowledge and tireless guidance have been crucial to the completion of my research.

I also wish to acknowledge a number of people for their assistance during the various experimental phases of my work. Bob Creber, Chris Fletcher, and Bill Marn of SPAWAR Systems Center, San Diego provided invaluable insight and training on the operation of the Seaweb system. RADM Steve Johnson (NUWC Newport), Peter Herstein (NUWC Newport), Rich Coupland (NUWC Newport), Giuseppe Izzi (NUWC Newport), Charles Humphrey (COMSUBPAC), and LCDR Steven Whear (COMSUBLANT) provided necessary guidance on the use of submarine navigation data.

My family provided wonderful encouragement and support throughout my studies. I especially thank my brother David for many general programming tips and his script to efficiently parse data from the Seaweb database which saved hours of manual transposition. Finally I would like to thank my friends and fellow students for making my time in Monterey an enjoyable learning experience.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The US Navy is increasingly using unmanned vehicles for operations in hostile environments. Examples of such missions include forward reconnaissance and mine clearing operations. Robust command and control (C2) infrastructure and precision guidance are vital to mission success when using these remote platforms.

Acoustic networks have been developed to facilitate C2 of unmanned underwater vehicles (UUVs). These networks can also be used to improve submarine communications at speed and depth for submarines (SSNs) operating nearby. Additionally, SSNs and UUVs operating with such a network can be used as intermittent gateways for fixed sensor assets in the network. In this role, the mobile node collects data from the network and then breaks the surface to transmit large data dumps to an over-the-horizon control center, eliminating the need for a continuous surface presence. Accurate knowledge of the position of a mobile node improves operation between assets in the network and increases network efficiency by minimizing the routing distance required for communications. Ranges calculated incidental to network communications provide a reliable fix source using already available modem hardware.

If ranges to the mobile node from multiple fixed nodes are collected, a position fix can be generated in a manner similar to a terrestrial receiver using GPS. Several algorithms for solving the position fix are compared using synthetic range data to determine their relative effectiveness. These tests also show that the mean and standard deviation of the resulting position error are linear functions of the standard deviation of the error in the measured ranges.

Operational testing with the Seaweb acoustic network, currently being developed by SPAWAR Systems Center, San Diego, was performed to collect range data to mobile nodes operating with the network. Data from these tests are analyzed and the resulting position fixes show a high degree of accuracy. An example of these test results is shown in Figure 1. The mobile node used for this experiment is a Slocum glider with a dead-reckoning system as its only navigation method available during submerged operations.

GPS fixes are received at the beginning and end of the track, and dead-reckoning positions are estimated when submerged. The fixes obtained from the network ranges show a significant improvement over the dead-reckoning system.

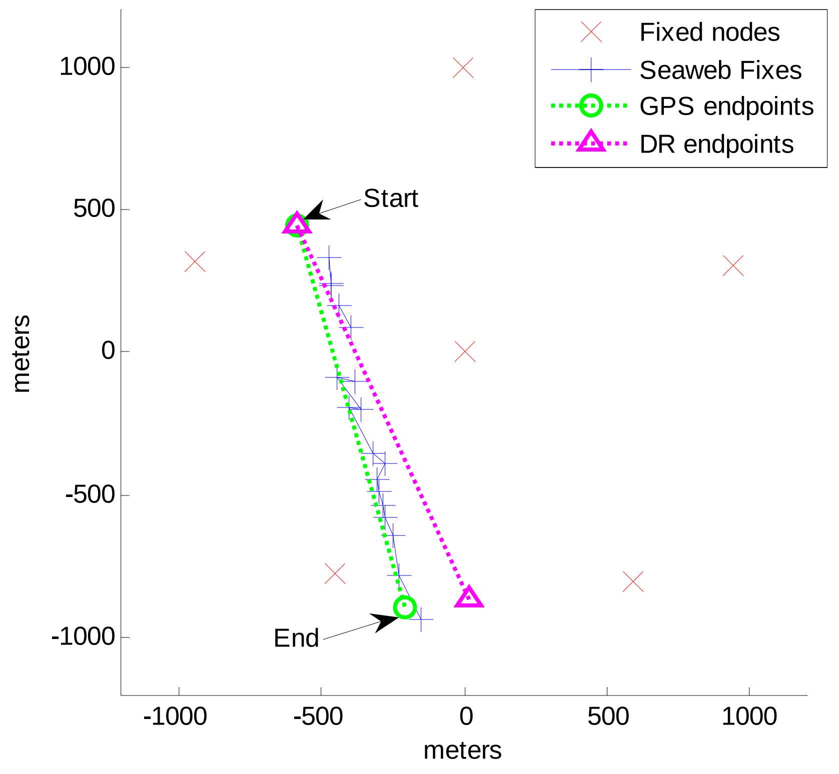

Figure 1. Slocum UUV track from July 21, 2005 showing initial and final GPS and dead-reckoning positions, over-plotted with Seaweb fixes

I. INTRODUCTION

A. MOTIVATION

Future U.S. Navy operations are expected to rely heavily on unmanned systems. For operations such as littoral surveillance or mine clearing, accurate navigation of these platforms is essential. This thesis examines the use of an acoustic network designed for communicating with such platforms as the basis for an underwater positioning system similar to the satellite based Global Positioning System. An algorithm robust enough to derive accurate positions from range measurements is needed for the successful operation of this system.

B. SEAWEB OVERVIEW

Seaweb is a system for underwater networked acoustic communications. A Seaweb network consists of an arbitrary number of sensor nodes, repeater nodes, and gateway nodes as described in [1]. Present implementations of Seaweb use Benthos Telesonar modems that are readily configurable to work with many mobile platforms including submarines and UUVs.

The Seaweb link-layer protocol includes a handshake operation from which the distance between the two communicating modems is calculated from the round-trip travel time. By obtaining simultaneous ranges between a mobile node and a number of fixed nodes, a Seaweb network can double as a portable acoustic tracking grid [2].

C. SEAWEB RANGING

The Seaweb handshake operation is illustrated in Figure 1.

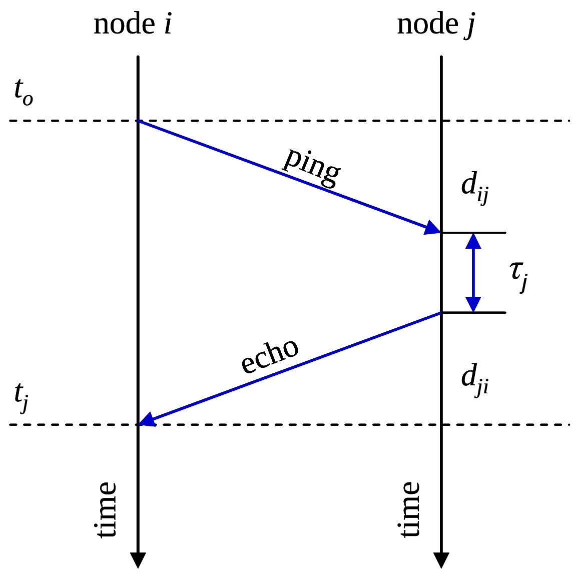

Figure 1. Seaweb ranging from node *i* to node *j* (from [2])

The round trip travel time is given by

$$t_j - t_0 = d_{ij} + \tau_j + d_{ji} \tag{1.1}$$

where d_{ij} and d_{ji} are the transmission times from node *i* to node *j* and from node *j* to node *i*, respectively, and τ_j is a random time delay inserted by the responding modem to decrease interference with other communications. Reciprocity dictates that $d_{ij} = d_{ji}$ and the one-way travel time is then

$$d_{ij} = \frac{t_j - t_0 - \tau_j}{2} \tag{1.2}$$

The range from node *i* to node *j* is the product of the one-way travel time and the speed of sound in seawater. The speed of sound is assumed to be a nominal value of 1500m/s yielding a node-to-node range in meters of

$$r_{ij} = 1500\left(\frac{t_j - t_0 - \tau_j}{2}\right) \tag{1.3}$$

The ranging calculations are performed by the initiating modem and ranges are logged in a database by the Seaweb server. There are several assumptions in this ranging technique. The most significant assumptions are the sound speed of 1500m/s and straight-line, direct path acoustic propagation. Despite these assumptions, range tests involving stationary nodes have shown that 97% of the ranges are accurate to within 10 meters.

In the broadcast ping process, one node transmits a utility packet addressed to all listening nodes in the network. The receiving nodes each respond with an "echo" utility packet which contain the data necessary to calculate the node-to-node range in equation (1.3). During the experiments analyzed in this thesis, broadcast pings from the UUVs were initiated by a command from the tending surface vessel sent through the gateway node. Range data were calculated from the echoes by the modem on the UUV and these data were transmitted through the network and logged on the Seaweb server on the tending vessel. This process is illustrated in Figure 2.

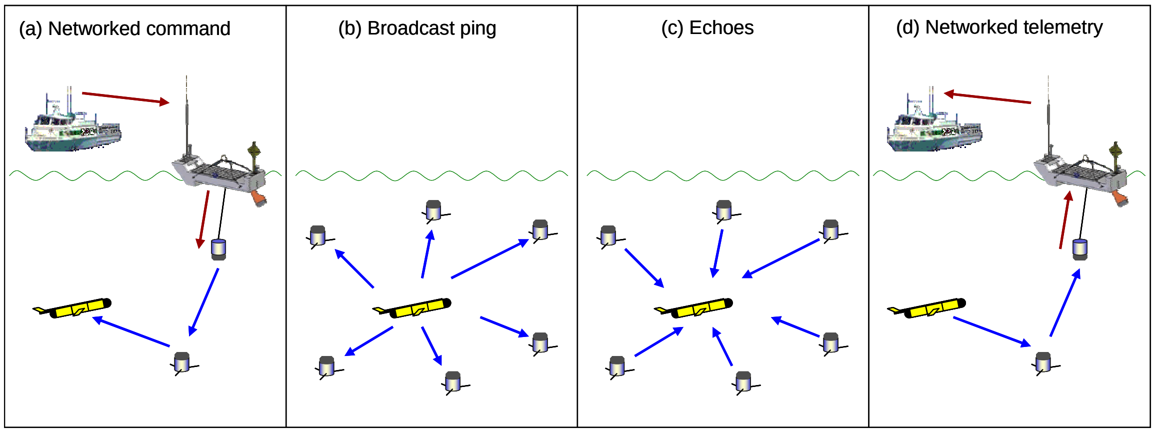

Figure 2. Broadcast Ping. The operator commands the UUV (a) to broadcast a ping (b). This elicits echoes from neighboring nodes (c). The UUV telemeters the calculated set of ranges back to the operator (d) (from [4]).

D. POSITION FIXING BY RANGING

In an *n*-dimensional coordinate system, the position of an object can be determined by measuring the distance between the object and *n*+1 known points as discussed in [3]. In the two-dimensional case, an arc is drawn centered at each of the known locations with a radius equal to the respective measured range as graphed in Figure 3. The position fix is shown by the intersection of the three arcs. Examples of this type of position fixing are radar navigation and the Global Positioning System (GPS).

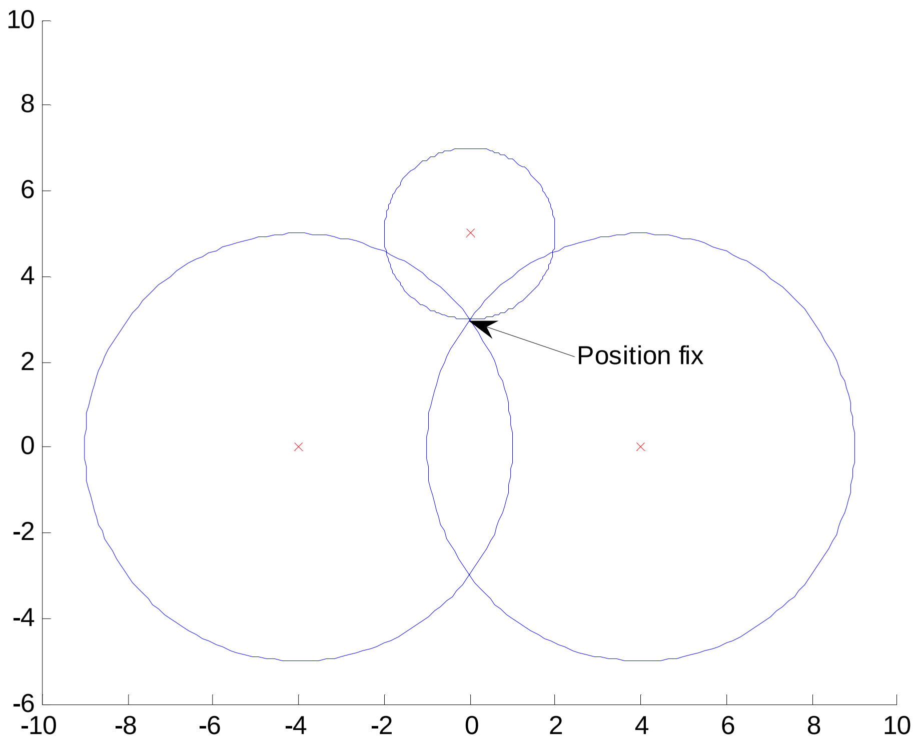

Figure 3. Two-dimensional position fix by ranging method.

Analytically, if the position of the *i*th known point is given by (x_i,y_i) and the measured range from the *i*th point to the object being fixed is given by r_i, the distance between the known point and the position of the object (*x*,*y*) is given by the Pythagorean relationship.

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2 \tag{1.4}$$

Since equation (1.4) is quadratic in two-dimensions, the position fix (*x*,*y*) is the solution to a system of *n*+1 simultaneous Pythagorean equations

$$\begin{bmatrix} r_1^2 \\ r_2^2 \\ r_3^2 \end{bmatrix} = \begin{bmatrix} (x-x_1)^2 + (y-y_1)^2 \\ (x-x_2)^2 + (y-y_2)^2 \\ (x-x_3)^2 + (y-y_3)^2 \end{bmatrix} = \begin{bmatrix} x^2 - 2xx_1 + x_1^2 + y^2 - 2yy_1 + y_1^2 \\ x^2 - 2xx_2 + x_2^2 + y^2 - 2yy_2 + y_2^2 \\ x^2 - 2xx_3 + x_3^2 + y^2 - 2yy_3 + y_3^2 \end{bmatrix} \tag{1.5}$$

Equation (1.5) is referred to as the fix equation and the unknown vector $[x\ y]^{\mathrm{T}}$ as the position fix.

In Figure 3 the ranges to the known points are assumed to be error free and there is a precise solution for the position fix. If the ranges do contain errors then a precise solution is not obtainable, however an area with high probability of containing the true position is indicated by the region surrounded by the intersections of the range circles as shown in Figure 4.

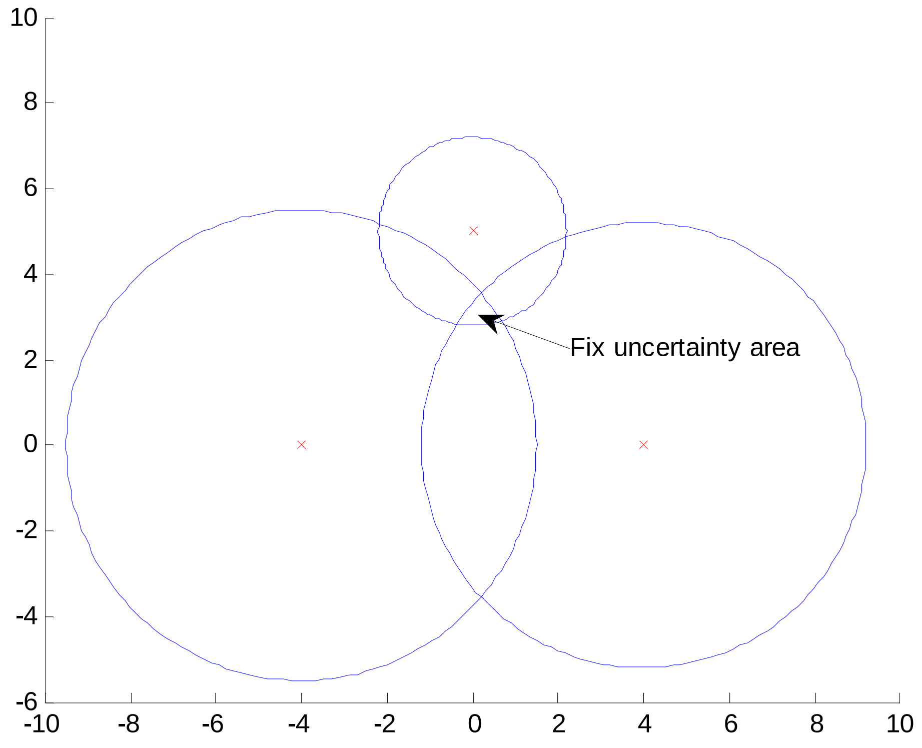

Figure 4. Two-dimensional position fix with imperfect range data.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND ON UNDERWATER POSITIONING

Previous use of Seaweb as a positioning system was based on solving for the intersections of pairs of range circles. This is equivalent to solving only the top two rows in equation (1.5). Because of the quadratic nature of the range circle equations, these pair-wise algorithms result in two solutions, one of which corresponds to the true position fix while the other corresponds to a reflection across the line connecting the two known points as shown in Figure 5.

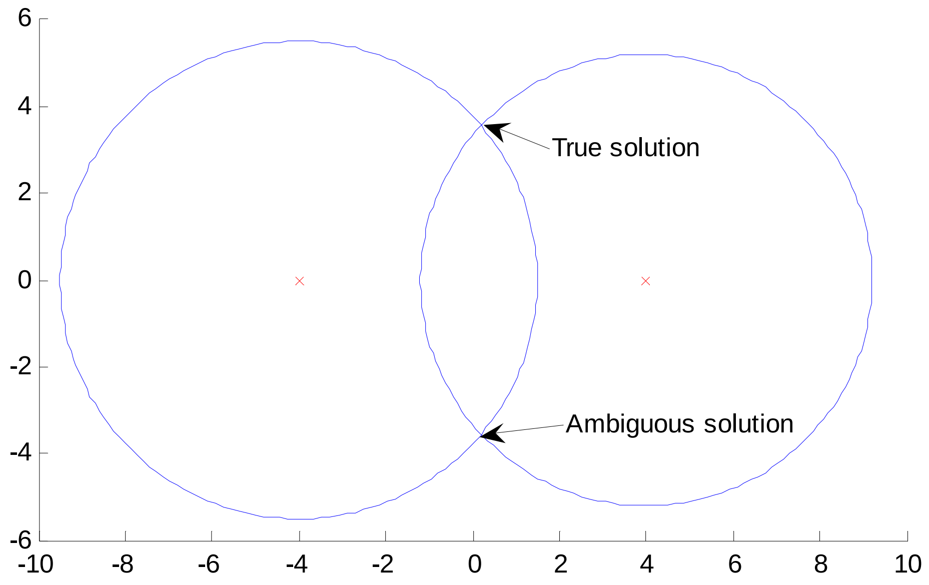

Figure 5. Ambiguous fix resulting from two range measurements

For N fixed nodes, this yields a total of $S = 2\binom{N}{2}$ solutions since each pair of fixed nodes will in general yield two intersections.

Two algorithms are implemented which compare these solutions to each other to eliminate the ambiguous estimates. Both of the methods are based on the premise that the intersections that correspond to the correct position will be concentrated around the true position while the ambiguous solutions will be significantly separated from this cluster.

A. WEIGHTING METHOD

In 2005 [2] created an algorithm in which a weighting value is assigned to each possible solution. The weighting value W_i for a particular solution $\boldsymbol{x}_i$ is based on the proximity of the solution to all other calculated solutions. Those solutions corresponding to the true position are assigned a higher weighting value. The ambiguous solutions are separated both from the true solution cluster and from the other ambiguous solutions and receive a low weighting factor. The final position $\boldsymbol{X}$ is then a weighted average given by

$$\overline{X} = \left(\sum_{i=1}^{S} W_i \overline{x}_i \right) \bigg/ S \tag{2.1}$$

B. CENTER OF MASS METHOD

Also in 2005, [4] tested an algorithm that attempts to eliminate the ambiguous solution by comparing both solutions from a given node pair to the overall mean position from all possible node pairs. For each fixed node pair, only the solution closest to the mean position is used to compute the final position.

C. DIFFERENCE LINEARIZATION METHOD

In [5], Krause formulates an algebraic solution to the GPS equations by a method called difference linearization in which the equation corresponding to a given satellite is subtracted from another equation in the system. This results in a set of linear equations that can be solved for the receiver position and time offset without the need for an iterative method. This thesis implements a similar method and compares performance to the pairwise weighting methods previously implemented.

III. DIFFERENCE LINEARIZATION METHOD

A. FORMULATION

As stated in Chapter II, previous work by Hahn and Ouimet solved the equations for pairs of intersecting range circles. This thesis evaluates the use of the difference linearization algorithm in [5] for solving the fix equation. This algorithm solves a simultaneous set of three range circle equations as given in equation (1.5) by reformulating the equality such that the squared unknown terms are eliminated. This leads to a pair of simultaneous linear equations in *x* and *y*

$$\begin{bmatrix} (r_1^2 - r_2^2) \\ (r_2^2 - r_3^2) \end{bmatrix} = 2\begin{bmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_2) & (y_3 - y_2) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} (x_1^2 - x_2^2) + (y_1^2 - y_2^2) \\ (x_2^2 - x_3^2) + (y_2^2 - y_3^2) \end{bmatrix} \tag{3.1}$$

with a solution of

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2}\begin{bmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_2) & (y_3 - y_2) \end{bmatrix}^{-1}\begin{bmatrix} (r_1^2 - r_2^2) + (x_2^2 - x_1^2) + (y_2^2 - y_1^2) \\ (r_2^2 - r_3^2) + (x_3^2 - x_2^2) + (y_3^2 - y_2^2) \end{bmatrix} \tag{3.2}$$

For simplicity of notation, let $\underline{x} = \begin{bmatrix} x \\ y \end{bmatrix}$, $P = \begin{bmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_2) & (y_3 - y_2) \end{bmatrix}$, and

$\underline{a} = \begin{bmatrix} (r_1^2 - r_2^2) + (x_2^2 - x_1^2) + (y_2^2 - y_1^2) \\ (r_2^2 - r_3^2) + (x_3^2 - x_2^2) + (y_3^2 - y_2^2) \end{bmatrix}$ so that equation (3.2) can be written as

$$\underline{x} = \frac{1}{2}P^{-1}\underline{a} \tag{3.3}$$

This method of difference linearization can be extended to an overdetermined system. In the case of a network with *N* fixed nodes the matrix *P* then has size (*N*-1 x 2) and the vector $\underline{a}$ has (*N*-1) elements. An overdetermined method of positioning was implemented as part of this thesis, but it was found to be less accurate than taking all available combinations of three nodes and averaging the solutions. One significant range error would skew the solution even if the other ranges were reliable.

In the combination method, individual solutions could be checked for accuracy and discarded if they did not pass. The combination algorithm also maintains P as a square matrix allowing exact solution by matrix inversion, in contrast to a least-squares estimation.

B. SOURCES OF RANGE ERRORS

There are several sources of error that can degrade the accuracy of the measured ranges, and hence lead to errors in the calculated positions. If a measured range $\hat{r}_i$ is corrupted by measurement error Δr_i then the measured range is $\hat{r}_i = r_i + \Delta r_i$ and vector $\underline{a}$ in equation (3.3) is estimated as

$$\underline{\hat{a}} = \begin{bmatrix} (r_1^2 + 2r_1\Delta r_1 + \Delta r_1^2 - r_2^2 - 2r_2\Delta r_2 - \Delta r_2^2) + (x_2^2 - x_1^2) + (y_2^2 - y_1^2) \\ (r_2^2 + 2r_2\Delta r_2 + \Delta r_2^2 - r_3^2 - 2r_3\Delta r_3 - \Delta r_3^2) + (x_3^2 - x_2^2) + (y_3^2 - y_2^2) \end{bmatrix} \tag{3.4}$$

Equation (3.3) will still have a mathematical solution when $\underline{a}$ is replaced with $\underline{\hat{a}}$, but this solution will not correspond to the true position when measurement errors become significant.

Because the range errors for each fixed node are independent and their effects on the position fix are nonlinear, it is difficult to perform a sensitivity analysis of a given positioning algorithm. In the simple case of only one range containing errors, say r_1, it can be shown that the position errors will be given by

$$\begin{aligned} \Delta x &= \frac{(y_3 - y_2)(2r_1\Delta r_1 + \Delta r_1^2)}{2\det(P)} \\ \Delta y &= \frac{(x_2 - x_3)(2r_1\Delta r_1 + \Delta r_1^2)}{2\det(P)} \end{aligned} \tag{3.5}$$

which shows that the position errors are also dependent on the fixed node geometry. Similar equations can be developed for errors in r_2 or r_3. The positioning algorithm uses range data from all available nodes such that if only a small portion of the available ranges are affected by significant range error, an accurate position can still be calculated from the uncorrupted ranges. Several case studies are presented in Chapters V and VI in support of a range error sensitivity analysis.

Another method of characterizing the position error is to determine the covariance of the position estimate $\underline{x}$ given by

$$Cov(\underline{x}) = E\left\{\underline{x}\underline{x}^T\right\} = \frac{\partial f(\underline{r})}{\partial \underline{r}} E\left\{\Delta\underline{r}\Delta\underline{r}^T\right\}\left(\frac{\partial f(\underline{r})}{\partial \underline{r}}\right)^T = \frac{\partial f(\underline{r})}{\partial \underline{r}} Cov(\Delta\underline{r})\left(\frac{\partial f(\underline{r})}{\partial \underline{r}}\right)^T \qquad (3.6)$$

where $\underline{r}$ is the vector of ranges and $f(\underline{r})$ is defined in (3.2) so that

$$\frac{\partial f(\underline{r})}{\partial \underline{r}} = \begin{bmatrix} \frac{\partial x}{\partial r_1} & \frac{\partial x}{\partial r_2} & \frac{\partial x}{\partial r_3} \\ \frac{\partial y}{\partial r_1} & \frac{\partial y}{\partial r_2} & \frac{\partial y}{\partial r_3} \end{bmatrix} = \frac{1}{\det(P)}\begin{bmatrix} r_1(y_3 - y_2) & r_2(y_1 - y_3) & r_3(y_2 - y_1) \\ r_1(x_2 - x_3) & r_2(x_3 - x_1) & r_3(x_1 - x_2) \end{bmatrix} \qquad (3.7)$$

An estimate of this covariance matrix is required if the Seaweb fixes are to be integrated into an onboard navigation system with a Kalman filter. For simplicity we assume that the errors in the range measurements are independent of one another but have the same standard deviation σ_r which gives

$$Cov(\Delta\underline{r}) = \sigma_r I \qquad (3.8)$$

Even for this simple case, the partial derivative matrix given by equation (3.7) shows that the covariance of the position error is dependent on both the network layout and the range to the vehicle. In reality there may be environmental or system features which result in different standard deviations for each node or which might introduce coupling in the range errors from different nodes that would lead to non-zero terms in the off-diagonal elements.

1. Neglecting Depth

While the difference linearization algorithm can easily be extended to three dimensions, the errors present in the ranges can cause significant errors in the solution of the depth coordinate. This is due mainly to geometric dilution of precision effects (discussed in section C of this chapter) as the fixed nodes are very nearly coplanar in the z-dimension. Even with minor variations in bathymetry over the network layout, the variation of fixed node depth is much smaller than the spread in x and y. Because of this, we have chosen to neglect depth and solve only for the two-dimensional position. Ignoring depth obviously causes some error in the range measurements because the vehicle will not be in the same plane as the fixed nodes. The closer the vehicle (in x and y) to a given fixed node, the more the range to that node will reflect the difference in depth between the vehicle and the node. A simple illustration is provided in Figure 6. If the angle θ is less than 71.8°, then the range error between r and r_h caused by neglecting depth will exceed 5%. For a depth separation z between the vehicle and node of 100m, this corresponds to a minimum good range of approximately 350m. In most cases, however, if the vehicle is very close to one fixed node such that the range is affected by depth, it will be far enough away from the other nodes in the network to minimize this effect for the remaining nodes.

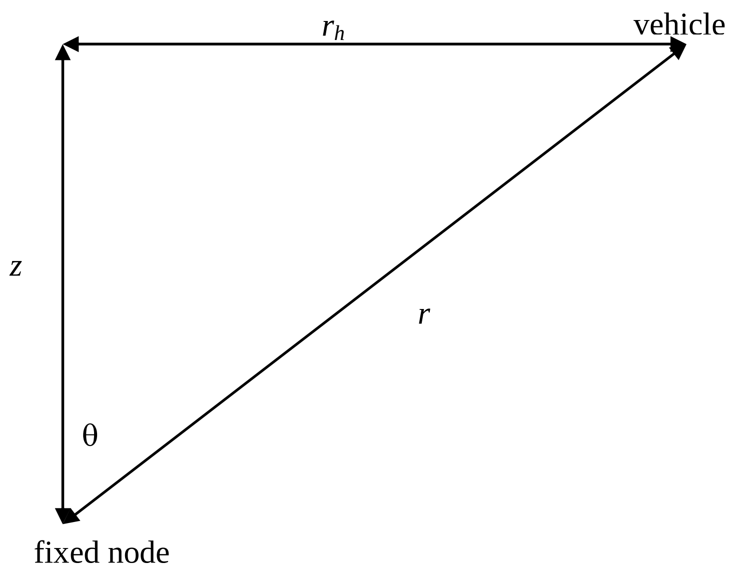

Figure 6. Effect of depth on vehicle range

Future network implementations can eliminate this source of range error by including depth sensors at all nodes.

2. Speed of Sound and Refraction

If the speed of sound estimate of 1500m/s used in equation (1.3) is not accurate, the calculated ranges will have corresponding inaccuracies. Sound speeds in the ocean typically range from 1480 to 1520 m/s or 1.3% from the assumed value [6]. Ranges calculated with an incorrect sound speed will have the same percentage error as the mismatch in sound speed. Also, sound speed often varies with depth, which leads to a refractive ray-path. The length of this refractive path will be longer than a straight-line path between the communicating nodes.

3. Multipath

Multipath error can arise if the channel geometry and sound speed profile allow for multiple ray-paths such as a surface or bottom reflection in addition to the direct path. The direct path is usually the shortest range and the most accurate. In most geometries of interest it should also have the highest SNR of the several path lengths. While the modems utilize a peak-detector filter which should eliminate most multipath effects, it is possible that a multipath response may actually have a higher SNR than the direct path and thus be inadvertently used for the range calculation.

4. Vehicle Motion

A vehicle in motion will travel some distance between the time it issues the broadcast ping request and the time it receives the response. The worst case for this type of error occurs when the vehicle is moving directly towards or away from a node. For example, a 5kt vehicle 1500m away from a fixed node would have a motion induced error of 2.57m or 0.17% [3]. Also, the responses are not normally received precisely simultaneously. Because of this, the vehicle will be at a slightly different position at the receipt of each ping response. This effect should be minimal if the responses are received within a few seconds of each other.

C. GEOMETRIC DILUTION OF PRECISION

The error in the calculated position is dependent on both the errors in the range measurements and the geometry of the fixed nodes. The influence of the fixed node geometry is a phenomenon known as geometric dilution of precision (GDOP). If each range arc is represented as an annulus to model the uncertainty in the measured range, the intersection of these annuli is then an area of likely target position. If the angle θ_{tr} is defined with vertex at the intersection of the annuli and rays to the center of each annulus, then as θ_{tr} becomes very small, the GDOP effects become more pronounced. The effects of GDOP are minimized when θ_{tr} is 90°. This is illustrated in Figure 7.

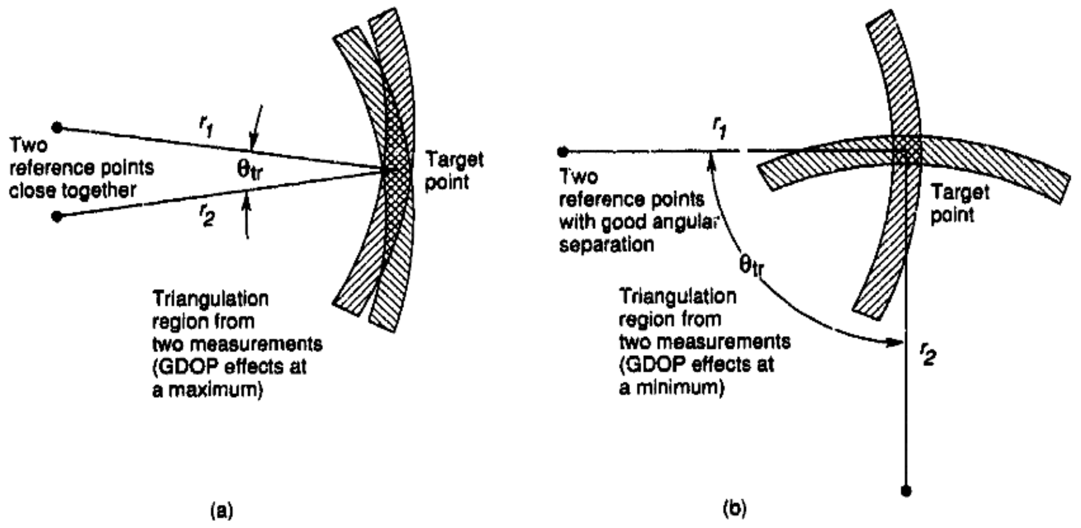

Figure 7. Geometric Dilution of Precision (from [7])

When using range data from three nodes the limiting worst case of GDOP occurs when all of the fixed nodes are collinear. In this case it is impossible to determine which side of the line corresponds to the true position. Analytically this is because the matrix *P* from equation (3.3) given by $\begin{bmatrix} (x_2 - x_1) & (y_2 - y_1) \\ (x_3 - x_2) & (y_3 - y_2) \end{bmatrix}$ is singular for the collinear case. Since the GDOP characteristics of the network are only dependent on the geometry of the fixed nodes, regions of high GDOP can be minimized during the network layout phase. A measure of the GDOP characteristics can be found by calculating the condition number of the *P* matrix [8]. A condition number near one indicates good GDOP characteristics while a large condition number indicates a nearly singular matrix and, as a result, poor GDOP characteristics.

IV. ALGORITHM IMPLEMENTATION

A. PROGRAM ORGANIZATION

The position fixing methods are implemented in Matlab. The program is modular for readability, maintainability, and applicability to both simulated and actual measured data.

1. Input and Data Acquisition

Input mechanisms are provided for reading fixed node positions (latitude and longitude) in either decimal degrees or degrees, minutes, and seconds. Ping ranges to these fixed nodes are also read from either text or Excel files. Inputting the data outside the positioning function allows for readier use with other fixing algorithms without reloading the data files.

2. Transformation from Latitude and Longitude to x-y

For locating the fixed nodes and comparison to other positioning systems such as GPS or onboard inertial systems, coordinates in latitude and longitude are transformed into a local coordinate system. Since the length of both a degree of latitude and a degree of longitude vary due to the shape of the Earth, these distances are specifically determined for the operating area according to equations provided by [9]. Once these lengths are locally determined for the operating latitude, a point such as one of the fixed nodes is chosen as the origin of the local coordinate system. The latitude and longitude of each point of interest are subtracted from those of the origin point and a simple unit conversion based on the calculated degree lengths is performed to obtain coordinates in meters relative to the chosen origin.

3. Position Determination

Since this algorithm solves the set of three simultaneous equations given by equation (1.5), there are actually $\begin{pmatrix} N \\ 3 \end{pmatrix}$ positions calculated for each broadcast ping, where N is the number of fixed nodes. The program does not assume a specific number of fixed nodes, so the value of $\begin{pmatrix} N \\ 3 \end{pmatrix}$ is calculated at the start of the program as a loop control parameter. The combinations themselves are also generated according to the algorithm in [10] for use as subscripts that index the specific node positions and ranges for each instance of calculation. With a given set of node positions and ranges as inputs, the matrix and vector given in equation (3.2) are formed and the equation solved for $\left[x\ y\right]^{\mathrm{T}}$. The position is then verified by comparing back-calculated ranges from the calculated position to the input ranges. If these back-calculated ranges differ from the input ranges by more than 10% of the input ranges, the solution for that set of nodes is considered invalid and ignored. When a solution has been calculated for each combination of fixed nodes, the valid solutions are averaged to determine a final position for that time instance. In addition to calculating a solution for each combination of fixed nodes, a solution of the overdetermined difference linearization equation described in Chapter III is also calculated in order to compare the relative accuracy of these two methods. For further comparison, the weighting method and center of mass method are calculated.

4. Output

The output of the program is normally a chartlet that shows the position of each fixed node and the final positions calculated for each time instance. These plots are performed in a different function so that tracks from other position sources or calculated by other algorithms can be overplotted for comparison.

V. ALGORITHM TESTING WITH SYNTHETIC DATA

A. ASSUMPTIONS AND SETUP

In July 2005 ranging experiments were conducted in Monterey Bay using a Slocum glider UUV as a mobile node within a Seaweb network [4]. The fixed grid geometry in these experiments is used as the grid in the following simulations. The recorded GPS positions and the associated x-y positions are shown in Table 1 and a chart of the node positions is given in Figure 8.

Node	Latitude (Decimal Degrees)	Longitude (Decimal Degrees)	y Position Difference (meters)	x Position Difference (meters)
R10	36.70563	-121.96362	0.00	0.00
R11	36.71463	-121.96367	998.75	-4.47
R12	36.70837	-121.95308	304.06	941.79
R13	36.6984	-121.95702	-802.33	589.73
R14	36.69864	-121.96871	-775.69	-454.81
R15	36.70852	-121.97417	320.71	-942.68

Table 1. Fixed node positions for simulations and July 2005 Slocum test showing original latitude/longitude and positions in meters relative to center node

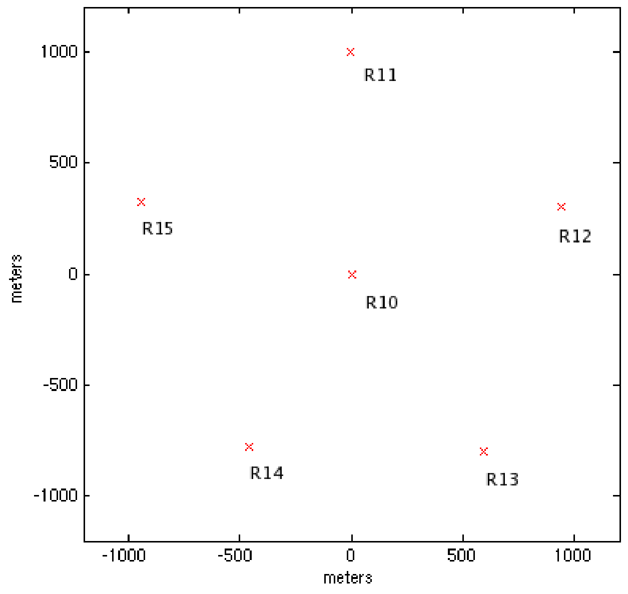

Figure 8. July 2005 grid geometry

The condition number of the *P* matrix for each combination of three nodes is shown in Table 2. As most of the condition numbers are between 1.5 and 2.1, this network shows good GDOP performance. However there are some sets such as nodes R10, R11, and R14 that give weaker GDOP performance. Detrimental effects from poor GDOP performance are minimized by discarding those solutions that fail to satisfy the original range equations within the 10% criteria discussed in Chapter IV.

Nodes	Condition Number of *P* matrix
R10, R11, R12	2.0290
R10, R11, R13	7.7187
R10, R11, R14	9.3952
R10, R11, R15	1.9820
R10, R12, R13	1.9859
R10, R12, R14	6.7679
R10, R12, R15	7.5646
R10, R13, R14	2.0441
R10, R13, R15	7.9865
R10, R14, R15	2.0839
R11, R12, R13	1.3928
R11, R12, R14	1.6504
R11, R12, R15	3.5279
R11, R13, R14	2.0178
R11, R13 R15	3.1235
R11, R14, R15	3.2134
R12, R13, R14	1.3576
R12, R13, R15	1.8224
R12, R14, R15	1.5818
R13, R14, R15	1.6182

Table 2. *P* matrix condition numbers for Monterey Bay Seaweb Grid

1. Data Characteristics

A mesh with spacing every 200m from (*x*,*y*) = (-1500, -1500) to (1500, 1500) resulted in 256 test points. Error-free ranges from each of the six nodes to each of these 256 points were calculated. The error free ranges were then corrupted with simulated measurement noise according to

$$\hat{r}_i = r_i + \Delta r_i \tag{5.1}$$

where Δr_i is Gaussian noise with mean zero and standard deviation σ_r (in meters). These corrupted ranges were then used as the input to the four different position fix algorithms (with minor changes to the programs from [2] and [4] to use the same ranges in all four methods). The calculated fix positions $(\hat{x}_i, \hat{y}_i)$ were compared to the true positions (x_i, y_i) of the 256 point mesh with the position error in meters given by

$$e_p = \left[\left(\hat{x}_i - x_i \right)^2 + \left(\hat{y}_i - y_i \right)^2 \right]^{1/2} \tag{5.2}$$

The entire mesh was tested in this manner 1000 times for each value of σ_r so that the resultant errors from each of the four algorithms could be compared.

B. CASE STUDIES

1. Data Set with Zero Range Error

To ensure the validity of all of the position fix algorithms a data set with zero range error was used as an initial case. All of the algorithms performed well in this case, with no errors greater then 0.5m. Even for this simple case, the center of mass method has significantly larger errors than the other methods. This indicates that for certain pairs of range circles the center of mass method is choosing the wrong solution. The mean and maximum errors from each algorithm are presented in Table 3.

	Combination method	Overdetermined method	Center of Mass Method	Weighting Method
mean error (m)	0.0039	0.0045	0.0126	0.0035
max. error (m)	0.0103	0.0134	0.4716	0.0167

Table 3. Position error statistics for zero range error data

2. Data Sets with Gaussian Range Error

Several case studies were performed for the case of Gaussian range error. The standard deviation values used were σ_r=4m, 15m, 30m, 50m, and 75m. Histograms of the position errors from each algorithm and each σ_r value are shown in Figures 9-13. As expected, the magnitude of the position errors increases with increasing range error, but from the histograms it can be seen that the maximum error from the two difference linearization methods is much less than that from the pairwise methods. The mean, standard deviation, and maximum errors, as well as confidence interval values for 50%, 75%, 90%, 95%, and 99% are given in tables 4-8 following each group of histograms. The statistics presented in the tables also validate the use of the difference linearization algorithm. In all of the tested cases, the mean, standard deviation, and maximum errors are smallest with the combination method of the difference linearization algorithm.

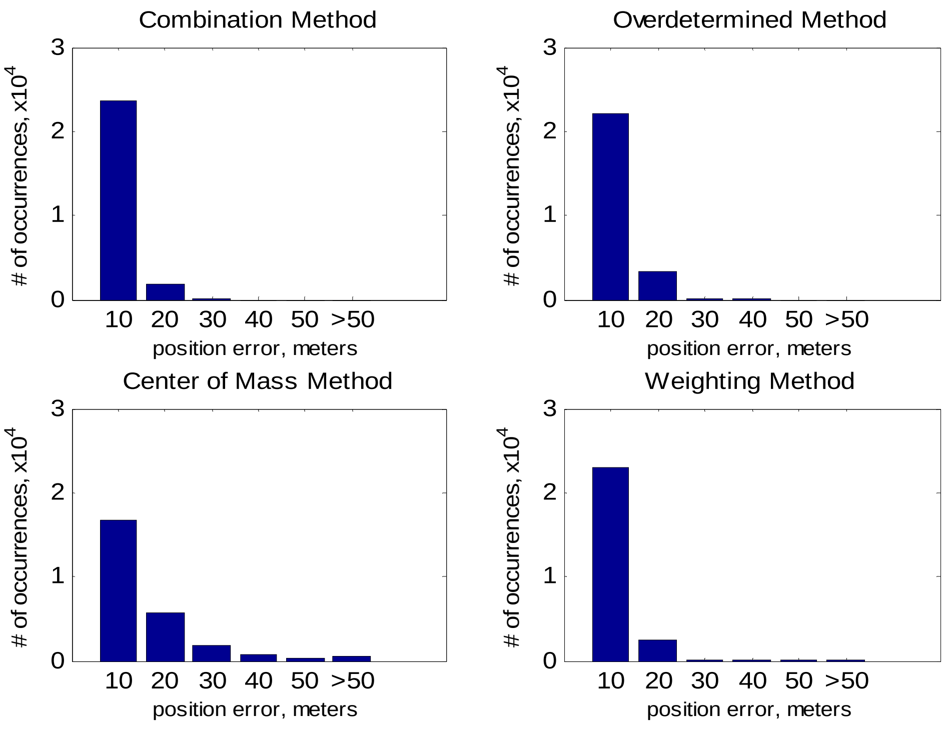

Figure 9. Position error histograms for σ_r=4m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	5.2	6.0	11.0	5.5
Error Std. Dev.	3.0	3.7	12.3	4.0
Max. Error	26.3	35.3	198.8	75.3
50% Error	4.7	5.4	7.2	4.7
75% Error	6.9	8.0	13.0	7.1
90% Error	9.3	11.0	23.3	10.2
95% Error	10.8	13.0	33.2	12.8
99% Error	14.1	17.3	64.8	19.9

Table 4. Position error statistics for σ_r=4m, all values in meters

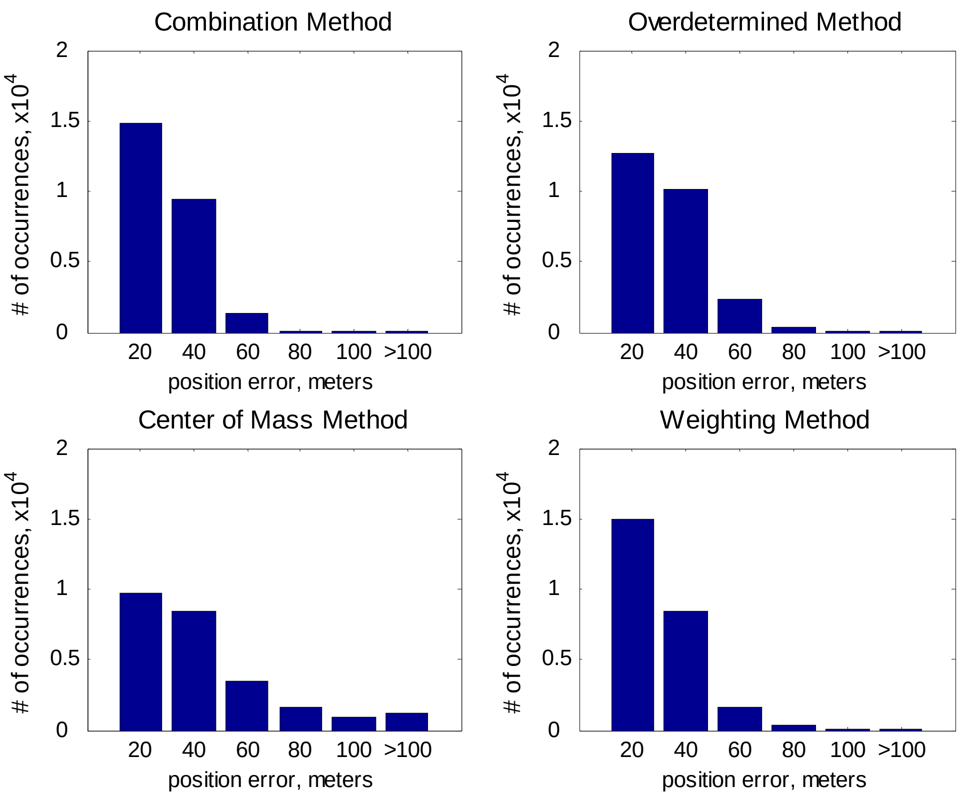

Figure 10. Position error histograms for σ_r=15m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	19.5	22.6	35.4	20.8
Error Std. Dev.	11.2	13.6	32.7	15.1
Max. Error	116.6	130.3	456.1	320.5
50% Error	17.7	20.2	25.5	17.5
75% Error	25.9	30.0	44.1	26.5
90% Error	34.7	41.0	74.0	38.2
95% Error	40.6	48.5	99.4	48.0
99% Error	52.7	64.8	164.9	75.2

Table 5. Position error statistics σ_r=15m, all values in meters

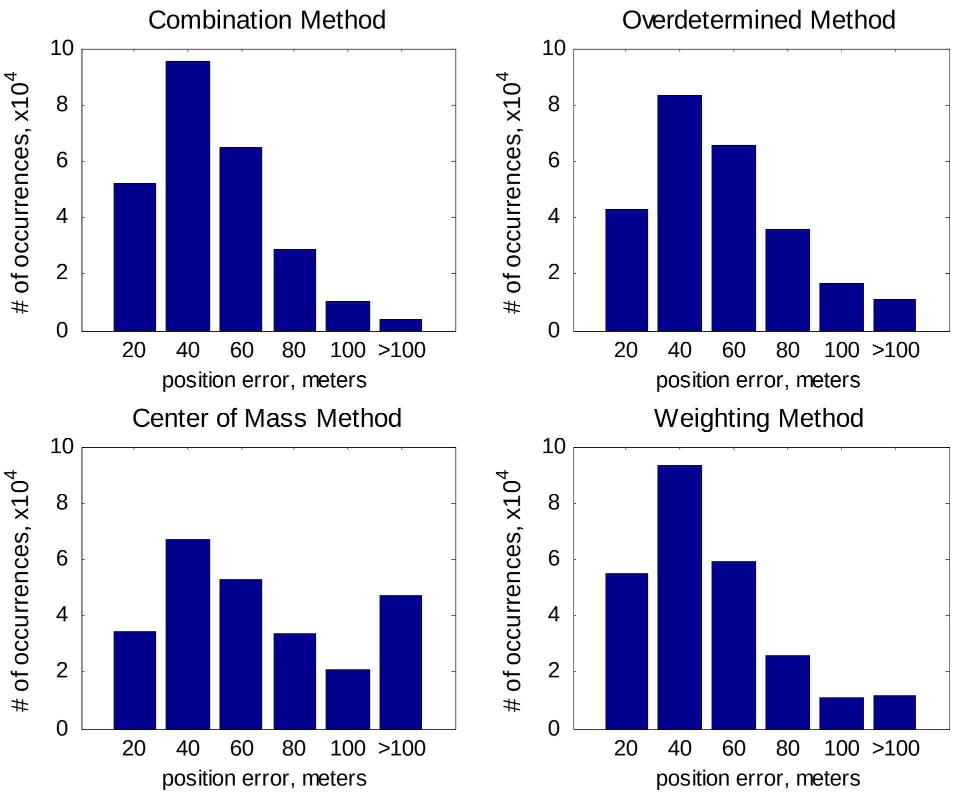

Figure 11. Position error histograms for σ_r=30m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	39.2	45.3	65.3	42.0
Error Std. Dev.	22.4	27.3	55.5	31.4
Max. Error	186.9	273.1	813.0	868.5
50% Error	35.6	40.3	49.0	35.3
75% Error	52.0	59.9	83.0	53.2
90% Error	70.0	82.1	134.6	76.7
95% Error	81.4	97.2	175.4	96.7
99% Error	105.4	129.9	273.3	155.9

Table 6. Position error statistics σ_r=30m, all values in meters

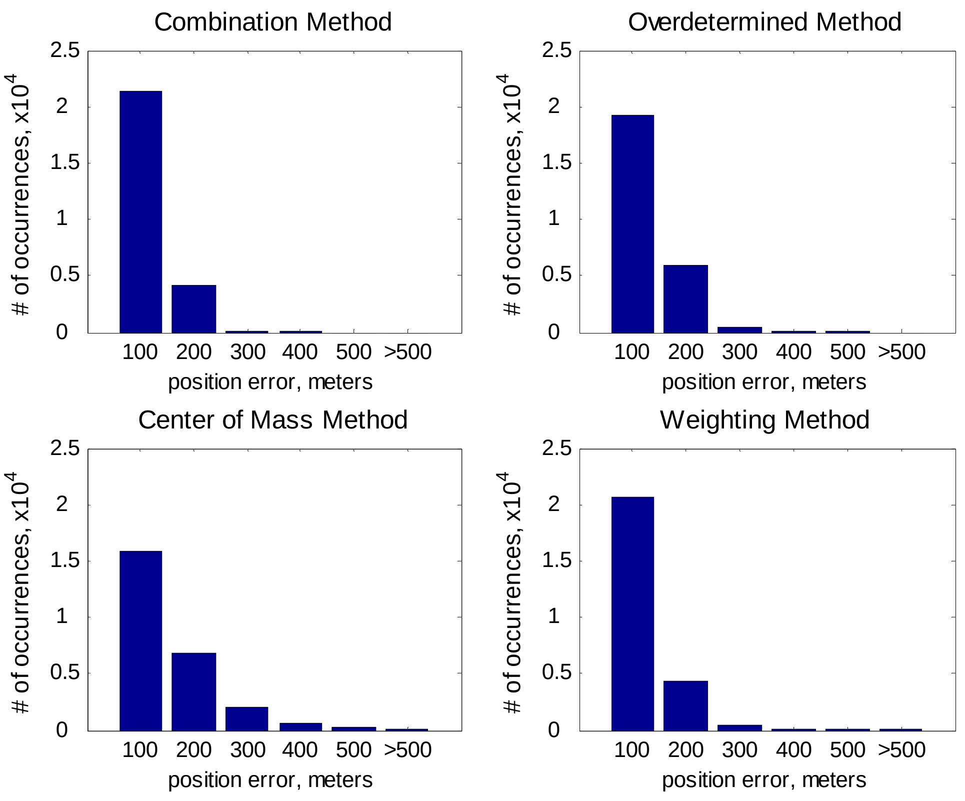

Figure 12. Position error histograms for σ_r=50m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	65.2	75.6	103. 2	71.3
Error Std. Dev.	37.5	45.6	83.7	56.6
Max. Error	317.9	420.0	1260.1	1603.3
50% Error	59.1	67.3	79.5	58.9
75% Error	86.6	100.1	132.1	89.7
90% Error	116.0	137.1	209.7	130.0
95% Error	135.8	162.3	269.6	166.6
99% Error	176.9	216.2	409.9	275.2

Table 7. Position error statistics for σ_r=50m, all values in meters

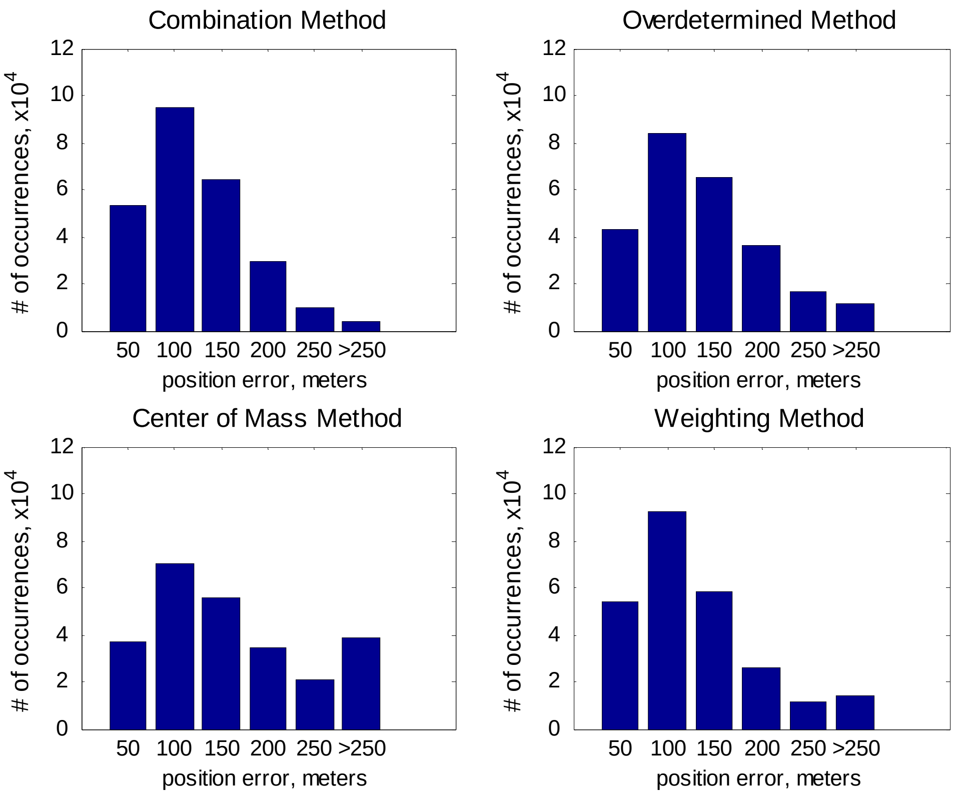

Figure 13. Position error histograms for σ_r=75m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	97.6	113.2	148.9	110.2
Error Std. Dev.	56.1	68.3	118.6	94.9
Max. Error	489.9	638.1	1958.2	2874.9
50% Error	88.5	100.7	116.6	89.1
75% Error	129.7	150.0	191.0	135.9
90% Error	173.9	205.1	298.6	200.8
95% Error	203.3	243.0	382.4	261.3
99% Error	264.4	324.7	575.5	470.9

Table 8. Position error statistics σ_r=75m, all values in meters

3. Data Set with Only Positive Range Error

A data set was also tested using Δr_i equal to the absolute value of a Gaussian distribution with σ_r=25m. The absolute value was used because most of the error sources discussed in Chapter III result in an overestimate of the range, the exception being underestimation of sound speed. Again the combination method of the difference linearization algorithm showed the best performance of the methods tested. Histograms of the error distribution are shown in Figure 14 with statistics in Table 9.

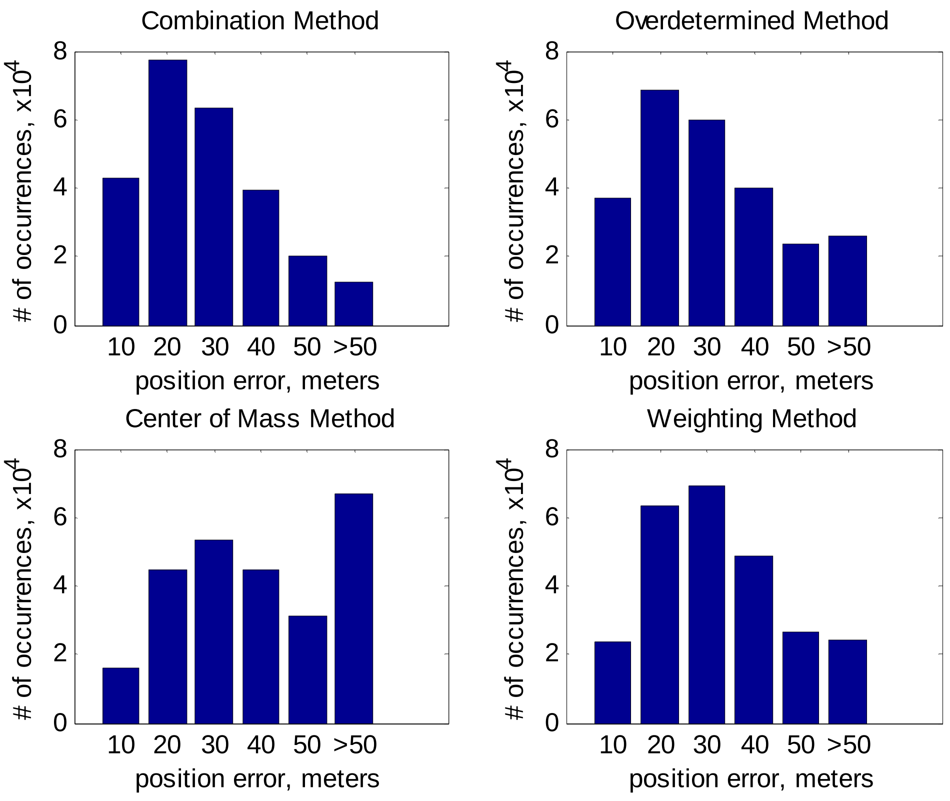

Figure 14. Position error histograms for positive Gaussian error, σ_r=25m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	23.5	26.9	41.3	28.5
Error Std. Dev.	14.0	17.5	32.6	17.1
Max. Error	123.6	200.9	454.5	360.9
50% Error	21.0	23.3	32.9	25.7
75% Error	31.7	36.0	51.0	36.7
90% Error	42.8	50.3	78.7	49.3
95% Error	50.0	60. 4	103.2	58.8
99% Error	64.3	82.4	168.1	84.1

Table 9. Position error statistics for positive Gaussian error, σ_r=25m, all values in meters

C. TESTING RESULTS

Throughout the range of σ_r values tested, the combination difference linearization method consistently performed the best of all four methods. Also, for the case of Gaussian range errors, both the mean and standard deviation σ_p of the position error were found to be linear functions of σ_r as shown in Figure 15. The slope of the line for mean error vs σ_r is 1.3 and the slope of the line for σ_p vs σ_r is 0.75. Both lines have a zero intercept. This result allows prediction of the position error if the distribution of input error is known.

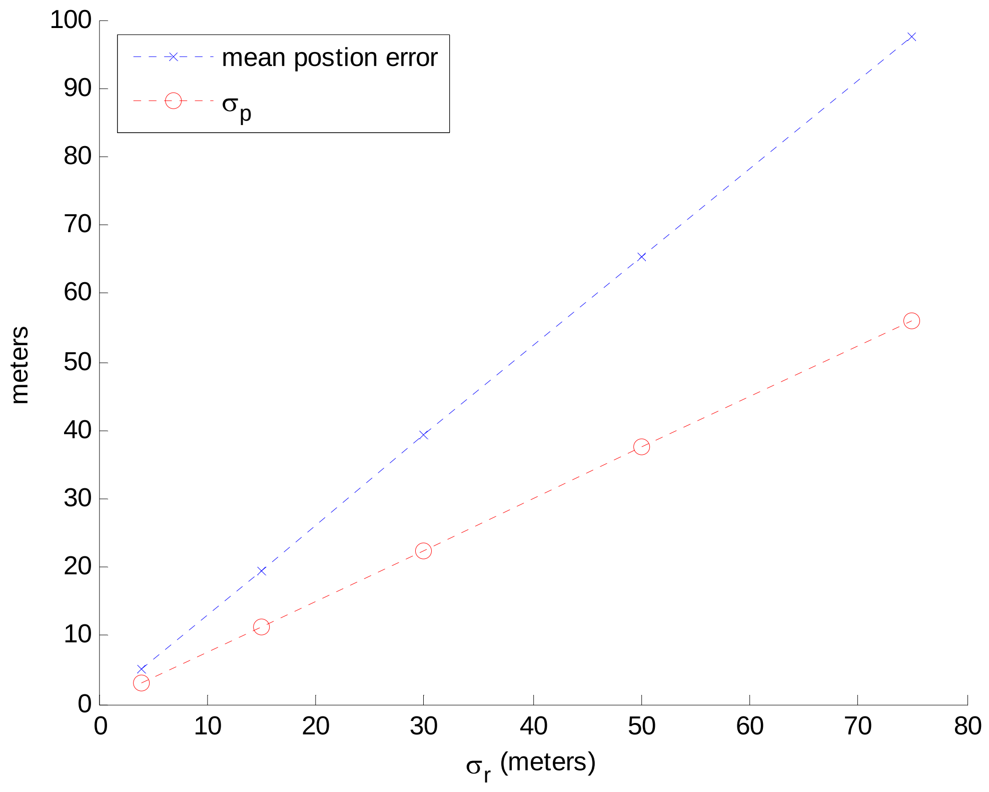

Figure 15. Mean and standard deviation of position error as a function of σ_r

VI. SYNTHETIC VEHICLE TRACKS

A. ASSUMPTIONS AND SETUP

Two simple tracks were simulated and tested in a similar manner to the mesh of known points tested in Chapter V. The first track consisted of a single leg from (-1200, 700) to (900, -1100). Fifty-one equally spaced points were calculated along the one leg defined by these waypoints. The second track had two legs defined by the waypoints (-1200, 700), (-900 -500), and (900, -1100). There were 101 equally spaced points along the second track, 50 per leg plus the final waypoint. The tracks are shown in Figure 16. Error-free ranges from each fixed node to each of the track points were calculated and corrupted with Gaussian noise for the simulations. As in Chapter V, 1000 iterations were performed for each noise value.

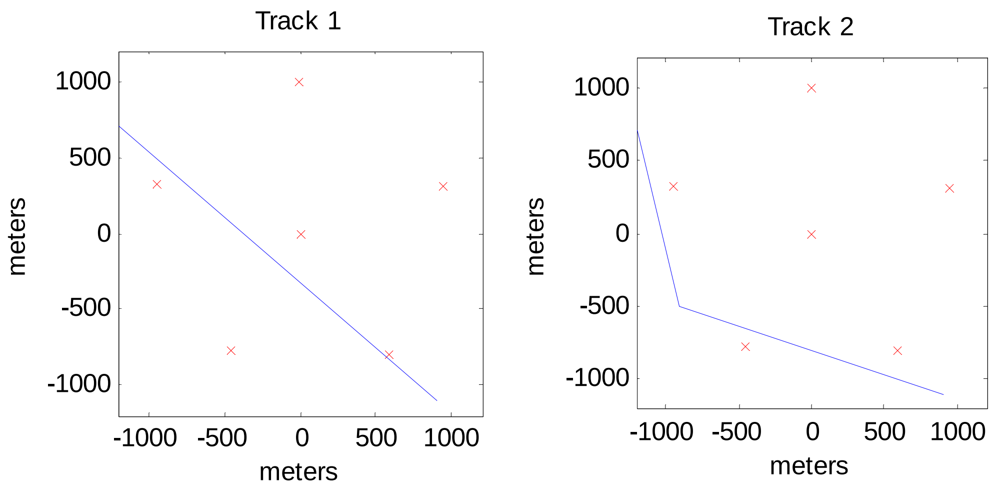

Figure 16. Synthetic tracks 1 and 2

B. CASE STUDIES

Two values of Gaussian error were added to the ranges for each track. The values of σ_r for these case studies were 4m and 50m. Position error histograms are shown in Figures 17-19 with error statistics in accompanying Tables 10-13. In all of the cases presented, the combination implementation of the difference linearization algorithm provides the best performance of the four methods.

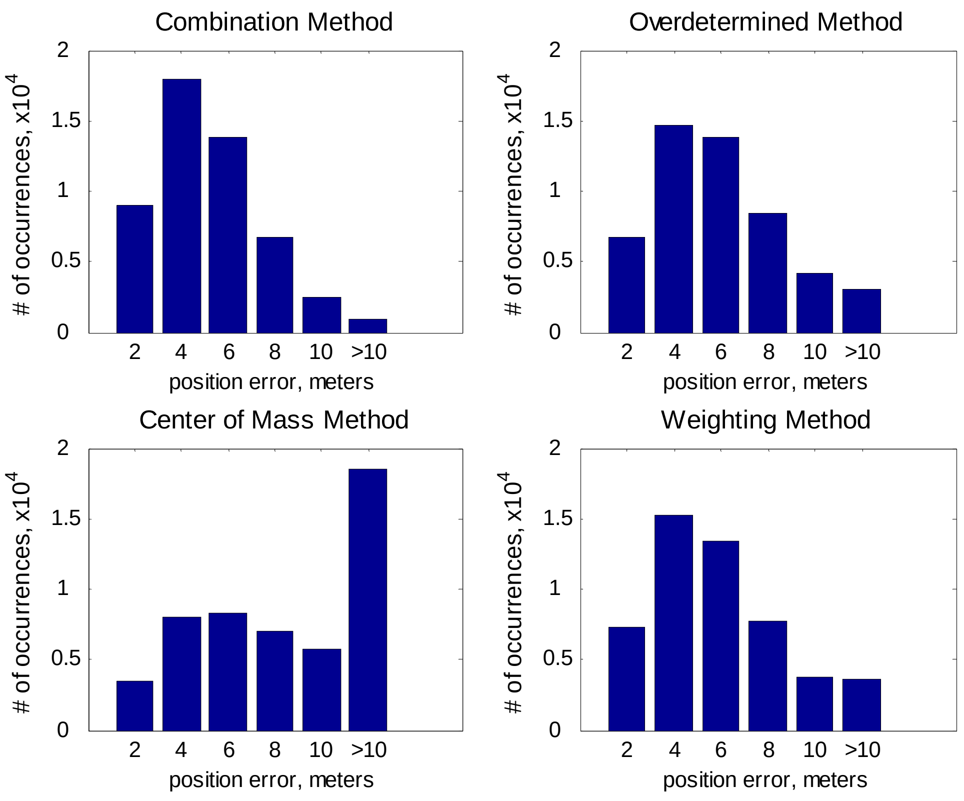

Figure 17. Position error histograms for single leg track with σ_r=4m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	4.2	5.0	9.5	5.1
Error Std. Dev.	2.3	2.9	7.2	3.5
Max. Error	18.5	22.8	69.6	62.7
50% Error	3.8	4.5	7.6	4.4
75% Error	5.5	6.6	12.8	6.5
90% Error	7.4	8.8	18.9	9.0
95% Error	8.5	10.4	23.4	11.0
99% Error	11.0	13.9	34.4	17.1

Table 10. Position error statistics for single leg track with σ_r=4m, all values in meters

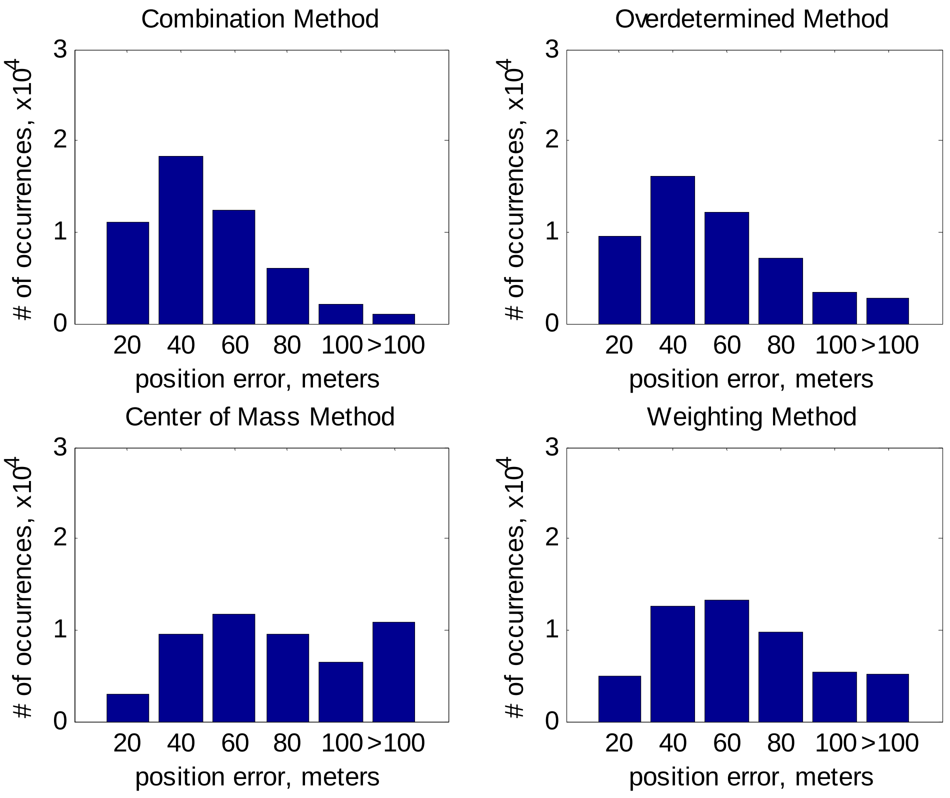

Figure 18. Position error histograms for single leg track with σ_r=50m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	39.4	45.6	72.1	58.0
Error Std. Dev.	23.4	29.2	44.9	36.6
Max. Error	210.5	267.2	421.3	519.6
50% Error	35.3	39.8	62.3	51.6
75% Error	52.9	61.0	92.7	74.4
90% Error	71.3	84.8	131.7	100.4
95% Error	83.6	101.4	159.6	121.3
99% Error	108.5	137.3	221.1	185.8

Table 11. Position error statistics for single leg track with σ_r=50m, all values in meters

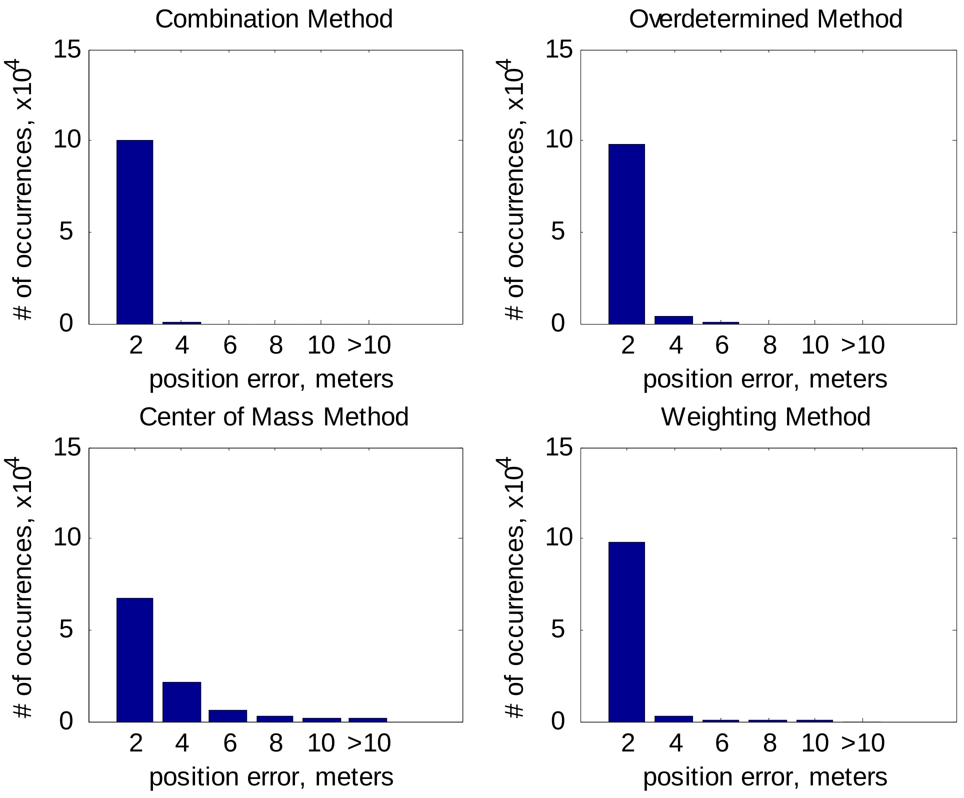

Figure 19. Position error histograms for two leg track with σ_r=4m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	3.6	4.2	10.2	4.4
Error Std. Dev.	2.1	2.6	10.4	2.4
Max. Error	17.3	22.3	113.0	40.4
50% Error	3.3	3.7	6.9	4.0
75% Error	4.9	5.6	12.3	5.7
90% Error	6.5	7.7	21.6	7.5
95% Error	7.6	9.2	31.0	8.8
99% Error	9.8	12.2	54.2	11.8

Table 12. Position error statistics for two leg track with σ_r=4m, all values in meters

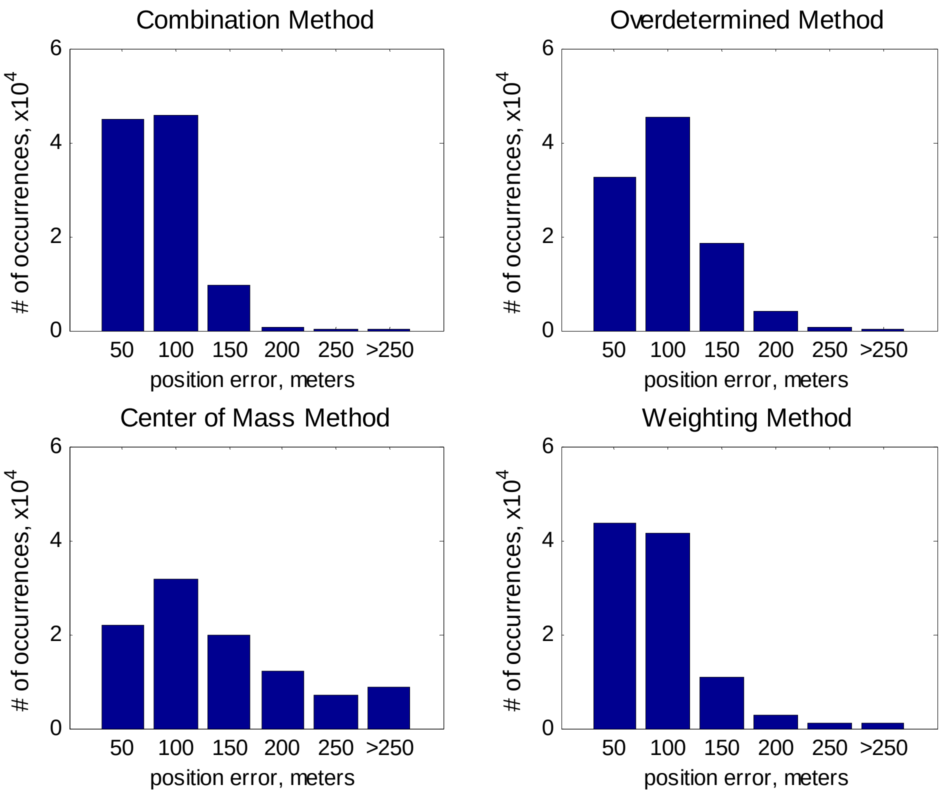

Figure 20. Position error histograms for two leg track with σ_r=50m

	Combination Method	Overdetermined Method	Center of Mass Method	Weighting Method
Mean Error	58.4	73.0	117.9	66.0
Error Std. Dev.	31.2	40.5	87.0	49.7
Max. Error	255.5	362.0	750.6	929.6
50% Error	54.4	67.2	94.7	55.9
75% Error	77.3	96.7	158.7	83.0
90% Error	100.7	127.5	236.0	117.4
95% Error	115.4	148.0	289.9	147.6
99% Error	145.8	190.6	405.1	256.6

Table 13. Position error statistics for two leg track with σ_r=50m, all values in meters

C. TESTING RESULTS

Although the statistical properties of the range values were the same, the mean and error covariance of the position errors for each set of cases were consistently smaller than those from similar tests in Chapter V. This is most likely due to the fact that the tracks do not cover the entire possible range of GDOP effects.

VII. AT SEA TESTING

A. JULY 2005 SEAWEB TESTING IN MONTEREY BAY

The July 2005 Seaweb testing used a Slocum glider UUV as a mobile node to be tracked within the fixed grid [4]. This vehicle is buoyancy driven and as such has significant depth excursions during operation as illustrated in Figure 21.

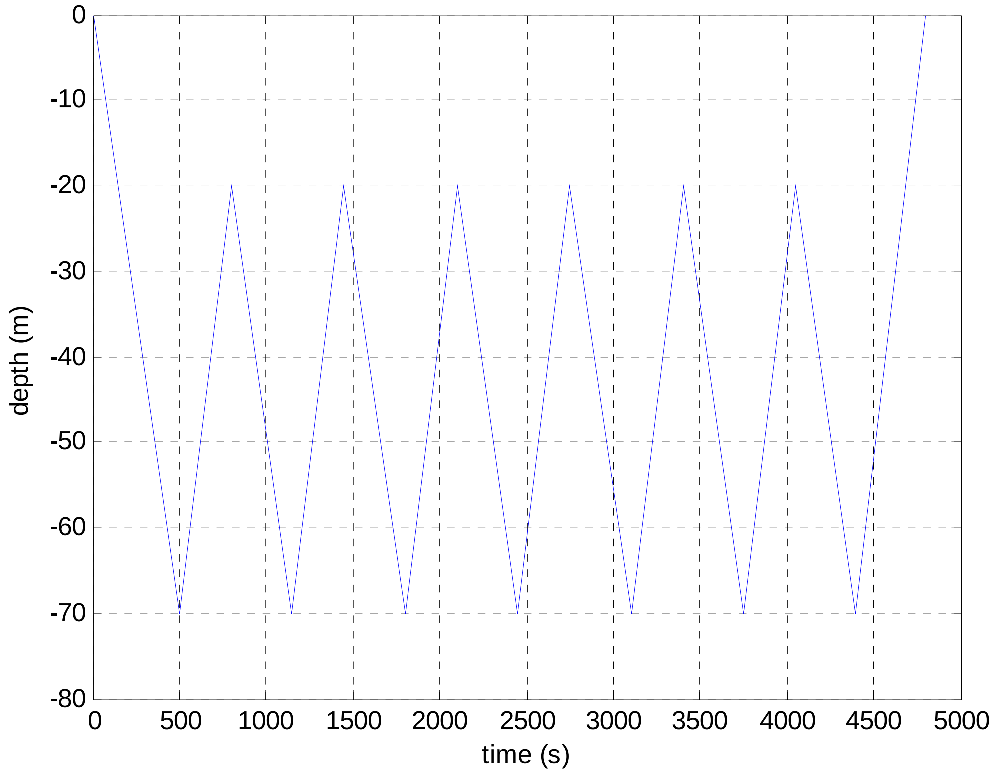

Figure 21. Slocum sawtooth motion (after [11])

A representative track from the Slocum experiment is shown in Figure 22. This figure shows the potential for navigation improvement obtained by utilizing Seaweb position fixes. Since the GPS, dead-reckoning, and Seaweb positions are not recorded for precisely the same times, an exact comparison is not possible, but it is apparent from the figure that the dead-reckoning positions did not match the GPS fix obtained by the vehicle when it returned to the surface. The final dead-reckoning position is 227.1m from the final GPS fix. The Seaweb fixes, however, appear to track well with the true vehicle motion, with the final Seaweb fix only 66.4m from the GPS endpoint. From the Seaweb fixes, it appears that the vehicle was attempting to follow the dead-reckoning track until it was pushed west by currents.

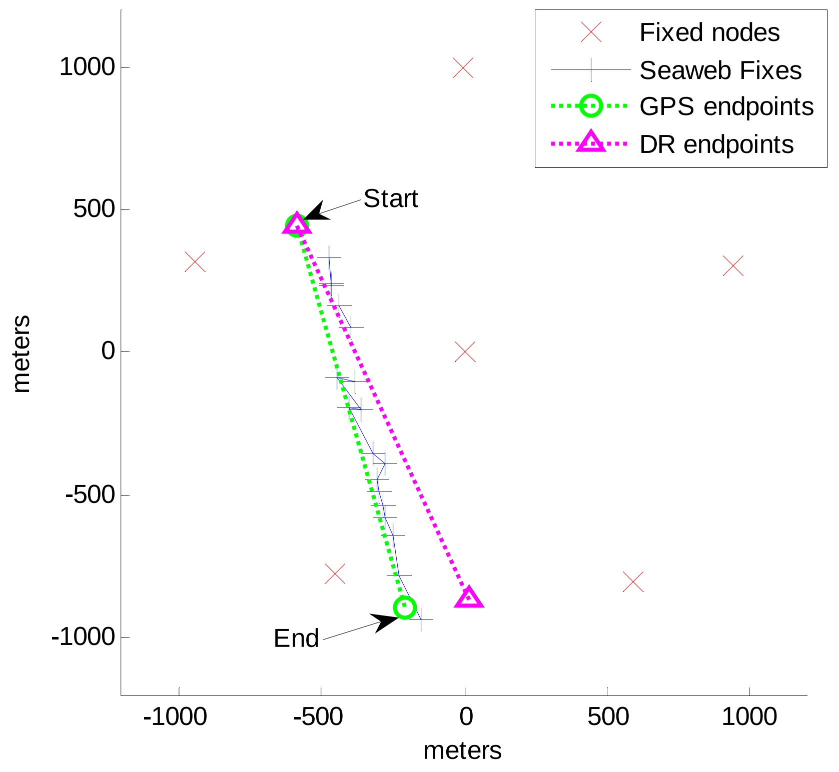

Figure 22. Slocum UUV track from July 21, 2005 showing initial and final GPS and dead-reckoning positions, over-plotted with Seaweb fixes

The ranging data used for positioning the Slocum are given in Table 14. During this event, 21 broadcast ping requests were issued with a possibility of 126 responses from the fixed grid and 420 positioning combinations for the track. Only 80 responses were actually received. The responses are grouped in such a way that there were 99 possible positioning combinations. Valid solutions are obtained from 59 of these 99 combinations. The remaining 40 combinations yield solutions that do not satisfy the 10% agreement criterion with the measured ranges.

Time	R10	R11	R12	R13	R14	R15
1119:36	579.6	827.5	1423.9	1563.0	NaN	503.1
1121:35	544.9	NaN	1416.3	NaN	NaN	501.3
1123:41	523.6	895.2	1413.4	1484.4	NaN	510.1
1125:47	518.7	NaN	1409.4	NaN	NaN	NaN
1128:35	504.6	NaN	1414.5	1420.2	979.5	NaN
1132:05	437.4	NaN	NaN	1331.4	882.1	607.5
1137:34	NaN	NaN	1415.8	NaN	784.0	NaN
1139:32	418.6	NaN	1416.3	1204.5	727.2	735.0
1141:26	435.3	NaN	1433.7	NaN	678.4	NaN
1144:05	453.1	NaN	1427.7	1133.4	643.8	NaN
1146:12	459.7	NaN	1439.5	NaN	595.3	NaN
1150:39	499.8	NaN	1455.0	1024.0	528.7	949.5
1152:44	541.2	NaN	NaN	991.5	488.2	NaN
1155:04	567.4	NaN	1484.1	971.1	455.8	1019.1
1157:05	NaN	NaN	1497.0	NaN	422.8	1065.7
1159:08	619.6	1581.1	1510.0	927.6	404.8	NaN
1201:24	667.9	NaN	1529.8	903.7	388.3	1137.1
1204:06	716.1	NaN	1551.9	865.3	372.4	NaN
1211:32	836.5	NaN	1617.9	829.2	361.8	NaN
1213:36	986.4	NaN	1663.9	808.8	386.2	NaN
1220:46	NaN	NaN	NaN	433.3	1480.9	NaN

Table 14. Range data for Slocum track

B. SUBMARINE TRACK RECONSTRUCTION

In October 2004, a submarine conducted tests with a Seaweb network. The range data from this exercise are analyzed to create position fixes that are then compared to the SSN’s Ring Laser Gyro Navigation (RLGN) position estimates. Calculated speeds from the RLGN data vary from 4.5 to 6.8kts, significantly faster than Seaweb tests with UUVs. A chart of this track is shown in Figure 23.

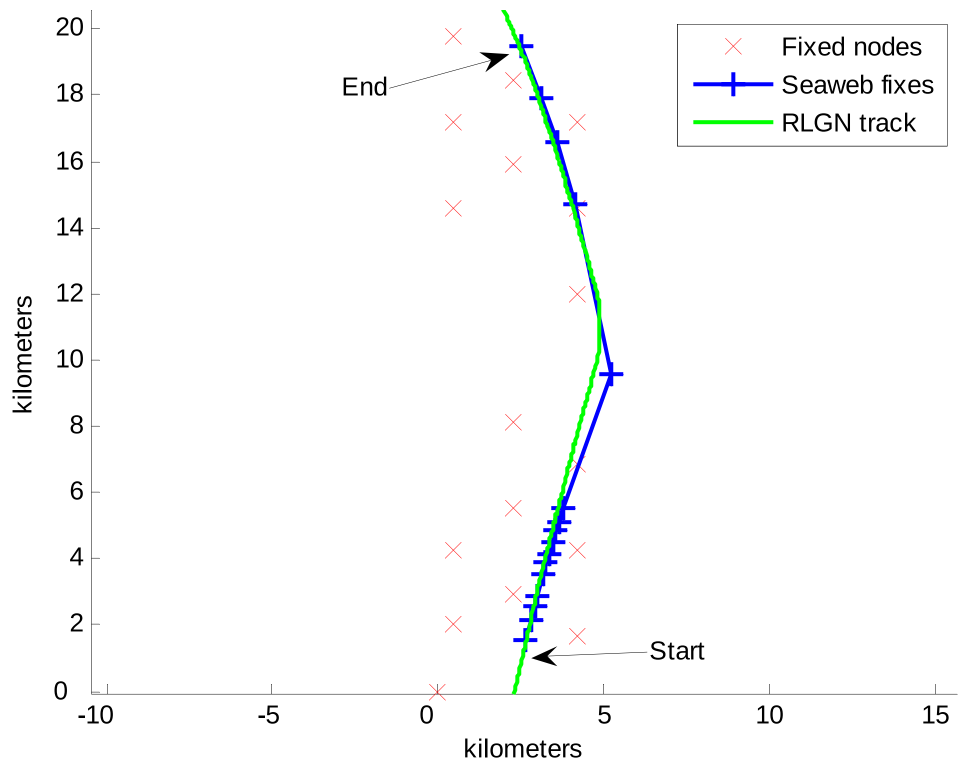

Figure 23. Submarine track reconstruction chart

As with the UUV experiments, another true fix source was not available for comparison during these tests. However 15 of the 16 Seaweb fixes fell within the standard position uncertainty radius of 1nm from the RLGN reading taken at the same time. The largest position difference was approximately 3350m, occurring at the course change in the track. The RLGN position for this time is further to the north than the Seaweb fix. This fix occurs on the edge of the network in an area with poor GDOP characteristics and as such is less likely to be accurate. The good performance while operating in the interior of the network does show the utility of using an acoustic network as an underwater positioning system.

During this event, 16 ping requests were sent and 55 total responses were received from 17 nodes. From these responses, 38 valid position fixes are generated.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

Despite the sometimes sparse data discussed in Chapter VII, the difference linearization algorithm and the available ranges consistently result in high quality fixes for an underwater positioning system. The linear relationship observed in Chapter V between the distributions of range error and position error is a good step in characterizing the quality of the position fixes.

B. RECOMMENDATIONS FOR FUTURE WORK

1. Test with Vehicle in Receipt of GPS

Although the simulations in Chapters V and VI indicate that the difference linearization algorithm performs better than previous fix methods, at-sea verification has been limited due to the fact that none of the vehicles tested were recording simultaneous fixes from a trusted source such as GPS. Comparisons to the vehicles' onboard inertial or dead-reckoning navigation systems have indicated that Seaweb ranges can be used to obtain a reasonable position fix. Experimental control with a vehicle simultaneously recording both GPS and Seaweb fixes would allow for determination of the absolute quality of the Seaweb fixes.

2. Algorithms that can Incorporate Asynchronous Ranges

Experiments to date have used a broadcast ping from the mobile node to all fixed nodes in range to obtain the position fix. As discussed in Chapter I, a range is calculated for each data transmission. These data transmissions are normally sent in accordance with a routing table resulting in ranges from only a few fixed nodes at a time as opposed to all nodes for a broadcast ping. In actual network operations, these data communications may occur more frequently than broadcast requests. An algorithm that could incorporate asynchronous range data from such communications would increase the utility of the positioning system. This would also address the ability to calculate position fixes in the cases of poor broadcast ping response when only one or two fixed node responses are received.

3. GDOP Weighting

As stated in Chapter III, the GDOP performance of a given set of nodes can be quantitatively assessed through the condition number of the *P* matrix of equation (3.3). A possible method for improving the overall algorithm would involve a weighting scheme based on the condition number of the associated node set. With this method, solutions that are expected to be good due to better GDOP characteristics would be more heavily favored than those from node sets with weaker GDOP performance.

4. Programming Inefficiencies

The combination algorithm is actually the slowest of the four methods presented, though still fast enough for real-time use. Run time is a function of several variables, but one of the most significant is the number of fixed nodes in the network, which determines the number of combinations for which solutions will be calculated. In a very large network it is likely that a mobile node will only be in range of a small fraction of fixed nodes at any given time. While enough data should still be available for obtaining accurate position fixes, it would not be necessary to attempt to calculate positions from the more distant nodes. An algorithm that uses the current position of the mobile node within the network to determine which fixed nodes to interrogate for ranges would improve the speed performance of the algorithm in large networks.

5. Kalman Filter

Integrating this algorithm with a Kalman filter would allow for prediction of the mobile node’s position and could also improve positioning accuracy. The motion model in the Kalman filter could be used to eliminate position fixes that do not make sense in the context of the mobile nodes track history or other physical constraints.

6. Integration with Other Navigation Sensors

The accuracy of the positioning system could also be improved by including data from other navigation sensors on board the platform, such as course, speed, and depth. Similar to the Kalman filter, these data would improve the algorithm’s ability to predict a future position and compare this prediction with calculated fixes.

7. Modeling and Testing for Error Estimation

The synthetic data presented in Chapters V and VI give a prediction of position error when the statistics of the range errors are known. In practice, these range errors are difficult to predict. A model that could predict range errors as a function of sound speed, water depth, and other factors would increase the usefulness of the insights gained in the synthetic testing.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDICES

A. MATLAB PROGRAMS DEVELOPED FOR THIS THESIS

```
function [a b] = uuv_main
%
% Thesis Main Program for difference linearization algorithm, can be
run as
% a function for comparison to other algorithms.  Output matrix a is
% position fix matrix for combination method, b is position fix matrix
for
% overdetermined method.
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

global num_dims all_nodes num_nodes range_data;

% clear
% close all;

% get user input for node file, range file, and number of dimensions to
% trilaterate (2 or 3)
% node_file = input(['\nPlease enter the name of the file that contains
'...
%     'the fixed node positions. \n' 'The current working directory is
'...
%     pwd '/.\n\n'], 's');
% node_mode = input(['\nPlease enter the format for the node
position'...
%     'file. [1]\n'...
%     '1 - Degrees Minutes Seconds \n' ...
%     '2 - Decimal Degrees \n\n']);
% % default conditional goes here and loop to ensure 1 or 2 is entered
% num_dims = input(['\nPlease enter the number of dimensions for '...
%     'position fixing, 2 or 3. \n\n']);
% % default conditional goes here and loop to ensure 2 or 3 is entered
% range_file = input(['\nPlease enter the name of the file that
contains '...
%     'the ranging data. \n' 'The current working directory is '...
%     pwd '/.\n\n'], 's');
%
%
%
num_dims = 2;
%
% % over_determined = 0;
%
% % input node positions so
% %
% %             |node1_x, node1_y, node1_z|
% %             |node2_x, node2_y, node2_z|
% % all_nodes = |node3_x, node3_y, node3_z|
% %             |node4_x, node4_y, node4_z|
% %             |node5_x, node5_y, node5_z|
% %             |node6_x, node6_y, node6_z|
% %
%
% pos_file = input('Enter the output file name. \n', 's');

% % node_file = '../Dec92005/node_gps.txt';
% node_mode = 2;
% node_file = '../Dec92005/sean_nodes.txt';
% % node_file = '../Dec92005/taswex_node.txt';
% all_nodes = node_read(node_file, node_mode);
% if num_dims == 3
%     heights = [0 -8 0 2 -3 -33]';
%     all_nodes = [all_nodes heights];
% end

num_nodes = length(all_nodes(:,1));

% need to get data in form
% time_data = [timestamp(1), timestamp(2), ... timestamp(N)]
%
%              |range1(1), range1(2), ... range1(N)|
%              |range2(1), range2(2), ... range2(N)|
% range_data = |range3(1), range3(2), ... range3(N)|
%              |range4(1), range4(2), ... range4(N)|
%              |range5(1), range5(2), ... range5(N)|
%              |range6(1), range6(2), ... range6(N)|
%



% range_data = node_ranges;

% range_file = '../simulations/rng_grid_200.txt';
% waypoints, transposed to match output where each colums is an
[X,Y,(Z)]
% set
% sym_track_b = [-1000 -1000 20; -750 0 80; 750 0 80; 1000 1000 20]';
% sym_track_c = [-1000 -1000 50; -750 0 50; 750 0 50; 1000 1000 50]';
% range_file = '../Dec92005/Node4out_b.txt';
% all_data = dlmread(range_file, ' ');
% range_data = dlmread(range_file, ' ', 0, 3)';

% a = 1.05;
% err_mult = 1;   % + 0.025*randn(size(range_data));
% err_add = 10*(randn(size(range_data)));
% max(max(err_mult))
% min(min(err_mult))
% mean(std(err_mult))
%
% range_data = err_mult .* range_data + err_add;

% time_data = all_data(:,1:6);
% range_data = all_data(:,3:end)';
% range_file = '../Dec92005/taswex_data.xls';
% range_data = xlsread(range_file);
% range_data = (range_data(2:end, :))';

% Jul 21 track5
% all_data = xlsread ...
%     ('../Sean/Thesis Definite/Read-in excel
files/program_jul21_ping_ranges');
% range_data = all_data(86:106, 1:6)';

time_data = [1:length(range_data(1,:))];

% now calculate positions from all combinations of data at each time
step,
% then calculate a center of mass position by averaging over the
% combinations
%

% delta_time = time_diff(time_data);
set_mat = comb_gen(length(all_nodes(:,1)), num_dims + 1);
num_combs = combination(num_nodes, num_dims + 1);

mult_pos = [];
mult_error = zeros(num_combs, length(time_data));
for time_count = 1:length(time_data)

%    if over_determined
        nan_ranges = sum(isnan(range_data(:, time_count)));
        if (num_nodes - nan_ranges) < (num_dims + 1)
            best_pos_od(:,time_count) = NaN;
        else
                [best_pos_od(:, time_count), err_vect] = ...
                    veh_pos_od(range_data(:, time_count), all_nodes,
0);
        end
 %   else
        for set_count = 1:num_combs
            set_list = set_mat(set_count, :);
            range_set = range_data(set_list, time_count);
            node_set = all_nodes(set_list, :);
            [mult_pos(set_count, :, time_count), ...
                mult_error(set_count, time_count)] = ...
                veh_pos(range_set, node_set, 0);
        end % for set_count
            best_pos(:, time_count) =
(nanmean(mult_pos(1:end,:,time_count)))';
  %      end % if over_determined
end % for time_count

a = best_pos';
b = best_pos_od';

% figure;
% hold on;
% if num_dims == 2
%     plot(all_nodes(:,1), all_nodes(:,2), 'rx');
%     plot(a(:,1), a(:,2), 'b+-');
% %     plot(b(:,1), b(:,2), 'mo-.');
% %     plot(mult_pos(:,1), mult_pos(:,2), 'go');
% %     plot(sym_track_c(1,:), sym_track_c(2,:), 'k+--');
% %     plot(real_node6(1), real_node6(2), 'go');
%     axis equal;
% %     xlim([-1200 1200]);
% %     ylim([-1200 1200]);
% %     legend('Fixed Nodes', 'Mean Combination Fix', 'Overdetermined',
...
% %         'Individual Fix', 'location', 'Best');
% %     title('2-D estimation');
% else
%     plot3(all_nodes(:,1), all_nodes(:,2), all_nodes(:,3), 'rx');
%     plot3(a(:,1), a(:,2), a(:,3), 'b.-');
% %     plot3(b(:,1), b(:,2), b(:,3), 'mo-.');
% %     plot3(sym_track_c(1,:), sym_track_c(2,:), sym_track_c(3,:),
'k+--');
%     legend('Fixed Nodes', 'By Combinations', 'Overdetermined', ...
%         'Waypoints', 'location', 'Best');
%     title('3-D estimation');
% % title('Ranged positions for node 7 without gateway ranges');
% % xlabel('meters');
% % ylabel('meters');
% end


function b = node_read(a, format)
%
% gets fixed lat/long position from file a in either degrees, minutes,
% seconds (format = 1) or decimal degrees and converts to local
coordinate
% system in meters (format =2).  Origin of the local coordinate system
is
% the first node in the file.
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%
node_gps = dlmread(a);

switch format
    case 1
% use following two lines if data file is in degrees, minutes, seconds
% node_lat = dms2deg(node_gps(:,1), node_gps(:,2), node_gps(:,3));
% node_long = dms2deg(node_gps(:,4), node_gps(:,5), node_gps(:,6));
node_long = dms2deg(node_gps(:,1), node_gps(:,2), node_gps(:,3))
node_lat = dms2deg(node_gps(:,4), node_gps(:,5), node_gps(:,6))
    case 2
% use the fillowing two lines if data file is in decimal degrees
node_lat = node_gps(:,1);
node_long = node_gps(:,2);
end
```

```
[latlen, longlen] = lat_len(mean(node_lat));


node_lat_rel_d = node_lat - node_lat(1);
node_long_rel_d = node_long - node_long(1);

node_lat_rel_m = node_lat_rel_d * latlen;
node_long_rel_m = node_long_rel_d * longlen;

b = [node_long_rel_m node_lat_rel_m];


function [latlen, longlen] = lat_len(lat)
%
% taken from http://pollux.nss.nima.mil/calc/degree.html conversion to
% units other than meters also available there.
%
% Compute lengths (in meters) of degrees of latitude and longitude
based
% on latitude
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

%       Convert latitude to radians
        lat = deg2rad(lat);

%       Set up "Constants"
        m1 = 111132.92;     % latitude calculation term 1
        m2 = -559.82;       % latitude calculation term 2
        m3 = 1.175;         % latitude calculation term 3
        m4 = -0.0023;       % latitude calculation term 4
        p1 = 111412.84;     % longitude calculation term 1
        p2 = -93.5;         % longitude calculation term 2
        p3 = 0.118;         % longitude calculation term 3

%       Calculate the length of a degree of latitude and longitude in
meters
        latlen = m1 + (m2 * cos(2 * lat)) + (m3 * cos(4 * lat)) + ...
                (m4 * cos(6 * lat));
        longlen = (p1 * cos(lat)) + (p2 * cos(3 * lat)) + ...
                    (p3 * cos(5 * lat));


function b = comb_gen(n, k)
%
% b = comb_gen(n, k)
%
% Generates all combinations (order does not matter) of 1:n taken k
% elements at a time.  Each element is listed as a row of the output
% matrix.
```

```
%
% This algorithm is taken from Discrete Mathematics and Its
Applications,
% by Kenneth H. Rosen, Random House, New York, NY 1988 (first edition)
% p. 227
%
% uses:
%   combination()
%
% used by:
%   uuv_main()
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

num_combs = combination(n,k);
comb_vect = 1:k;        % first line
comb_mat = comb_vect;

for comb_count = 2:num_combs
    elem_count = k;
    while comb_vect(elem_count) == (n - k + elem_count)
        elem_count = elem_count - 1;
    end
    comb_vect(elem_count) = comb_vect(elem_count) + 1;
    count_2 = (elem_count + 1):k;    % Matlab will not create
decrementing
            % vectors unless the decrement is specified, ie 5:1 will
NOT
            % create [5 4 3 2 1] but 4:-1:1 would create [4 3 2 1].
Thus,
            % if elem_count >= k, count_2 will be an empty matrix and
the
            % following line indexing on count_2 will be ignored.
    comb_vect(count_2) = comb_vect(elem_count) + count_2 - elem_count;
    comb_mat = [comb_mat; comb_vect];
end
b = comb_mat;


function [b, error_out] = veh_pos(ranges, node_pos, error_in)
%
% [b, error] = veh_pos(range, node_pos)
%
% triangulate the cartesian coordinates of a point b in n dimensions by
% triangulating given the straight line distances (ranges) to the
unknown
% point from n+1 knownn locations (node_pos).
%
% The set of equations to be solved for (x,y) are of the form
%
% r1^2 = (x-x1)^2 + (y-y1)^2
%
% where r1 is an element of ranges and (x1,y1) are given by a row of
% node_pos).  These equations are solved by subtracting pairs of
equations
% which eliminates squared terms of the unknowns and leads to equations
of
% the form
%
% [x] = (2*(x2 - x1))\((r1^2 - r2^2) - (x1^2 - x2^2))
%
% (see thesis report for full derivation)
%
% since this method relies on the difference between positions, it can
% return a position even if the range circles don't intersect, so we
check
% the calculated position with range_check(), which solves for ranges
from
% the calculated position to the known locations.  range_check places a
'1'
% in the error vector if the position does not match with the measured
% ranges.  If any input range is zero or NaN, the output position is
% returned as a vector of NaN values and a '2' is placed in the error
% vector.
%
% uses:
%   sqr_diff(), pos_diff(), range_check(), nan_tester()
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

error_out = error_in;
        range_diff_sqr = sqr_diff(ranges);      % (r1^2 -r2^2)
        node_diff = pos_diff(node_pos);         % (x2 - x1)
        node_diff_sqr = sqr_diff(node_pos);     % (x1^2 - x2^2)
        node_diff_inv = pinv(2*node_diff);

        diff_sqr_vect = range_diff_sqr - node_diff_sqr;

        b = 2*node_diff \ (range_diff_sqr - node_diff_sqr);

        [b, error_out] = range_check(b, ranges, node_pos);
        if error_out ~= 0
             b = [NaN NaN];
        end % if range_error


function [b, error_out] = veh_pos_od(ranges, node_pos, error_in)
%
% [b, error] = veh_pos(range, node_pos)
%
% triangulate the cartesian coordinates of a point b in n dimensions by
% triangulating given the straight line distances (ranges) to the
unknown
% point from n+1 knownn locations (node_pos).
%
% The set of equations to be solved for (x,y) are of the form
```

```
%
% r1^2 = (x-x1)^2 + (y-y1)^2
%
% where r1 is an element of ranges and (x1,y1) are given by a row of
% node_pos).  These equations are solved by subtracting pairs of
equations
% which eliminates squared terms of the unknowns and leads to equations
of
% the form
%
% [x] = (2*(x2 - x1))\((r1^2 - r2^2) - (x1^2 - x2^2))
%
% (see thesis report for full derivation)
%
% since this method relies on the difference between positions, it can
% return a position even if the range circles don't intersect, so we
check
% the calculated position with range_check(), which solves for ranges
from
% the calculated position to the known locations.  range_check places a
'1'
% in the error vector if the position does not match with the measured
% ranges.  If any input range is zero or NaN, the output position is
% returned as a vector of NaN values and a '2' is placed in the error
% vector.
%
% uses:
%   sqr_diff(), pos_diff(), range_check(), nan_tester()
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

error_out = error_in;
% find and remove NaN's before creating pos_diff and sqr_diff
    nan_index = find(isnan(ranges));
    ranges(nan_index) = [];
    node_pos(nan_index,:) = [];
    if length(ranges) > 2
%
        range_diff_sqr = sqr_diff(ranges);      % (r1^2 -r2^2)
        node_diff = pos_diff(node_pos);         % (x2 - x1)
        node_diff_sqr = sqr_diff(node_pos);     % (x1^2 - x2^2)
        node_diff_inv = pinv(2*node_diff);

        diff_sqr_vect = range_diff_sqr - node_diff_sqr;



        b = 2*node_diff \ (range_diff_sqr - node_diff_sqr);

        [b, range_error] = range_check(b, ranges, node_pos);
        if range_error ~= 0
            error_out = [error_out, range_error];
        end % if range_error
%     % end % if nan_tester
```

```
    else
        b = NaN;
        range_error = 2;
    end



function b = pos_diff(a)
%
%
% returns a matrix that is the difference of the rows of the input
matrix.
% In general, if a has dimensions m x n, b will have dimensions (m-1) x
n.
%
%      |x1 y1|                           |(x2 - x1), (y2 - y1)|
% If a=|x2 y2| then pos_diff returns b = |(x3 - x2), (y3 - y2)|
%      |x3 y3|
%
% nb that pos_diff subtracts the upper row from the lower row, which is
% opposite from the operation of sqr_diff.  pos_diff requires that the
% input matrix a have at least 2 row but supports any higher number of
% dimensions.
%
% used by:
%   veh_pos()
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

b = a(2:end, :) - a(1:end-1, :);


function b = sqr_diff(a)
%
%
% returns a column vector of the differences of squared range or
position
% data.
%      |a1|                         |a1^2 - a2^2|
% If a=|a2| then sqr_diff returns b=|a2^2 - a3^2|
%      |a3|
%
%      |x1 y1|                         |(x1^2 - x2^2)+(y1^2 - y2^2)|
% If a=|x2 y2| then sqr_diff returns b=|(x2^2 - x3^2)+(y2^2 - y3^2)|
%      |x3 y3|
%
% nb that sqr_diff subtracts the lower row from the upper row, which is
% opposite from the operation of pos_diff.  sqr_diff requires that the
% input matrix a have at least 2 rows but supports any higher number of
% dimensions.
%
% used by:
%   veh_pos() in both forms above.
```

```
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

sqr_mat = a(1:end-1, :).^2 - a(2:end, :).^2;
b = sum(sqr_mat, 2);


function [pos_out, error] = range_check(pos_in, range, node_pos)
%
% b = range_check(position, range, node_pos)
%
% checks that the straight line differences between a calculated point
% (position) and a set of given fixed locations (node_pos) is equal to
% measured ranges (range) within an acceptable error
%
% returns 0 if the calculated range and measured range match
% returns 1 if the calculated range and measured range do not match
%
% uses:
%   veh_pyth()
%
% used by:
%   veh_pos()
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

acc_error = 0.1;    % acceptable error as a percentage of range
if abs((range - veh_pyth(pos_in, node_pos))./range) ...
        < acc_error
    error = 0;      % position and range values check
else
    error = 1;      % position and range values do not check
    for i = 1:length(pos_in(:,1))
        for j = 1:length(pos_in(1,:))
            if abs(pos_in(i,j)) == inf
                pos_in(i,j) = NaN;
            end % if a
        end % for j
    end % for i
end % if
pos_out = pos_in;

function b = veh_pyth(position, node_pos)
%
% b = veh_pyth(position, node_pos)
%
% uses the Pythagorean theorem to calculate the ranges from a point
% (position) to each of a set of fixed nodes with locations given by
the
% rows of node_pos.
```

```
%
%
%                |x|                 |x1 y1|
% if position = |y| and node_pos = |x2 y2| then veh_pyth returns
%                                    |x3 y3|
%
%          |(x - x1)^2 + (y - y1)^2|
% b = sqrt|(x - x2)^2 + (y - y2)^2| such that each row i of b is the
%          |(x - x3)^2 + (y - y3)^2|
%
% straight line range from (x,y) to the location (xi,yi) given in row i
of
% node_pos.  veh_pyth supports any number of dimensions and fixed
points.
%
% used by:
%   range_check()
%
% Mike Reed
% Naval Postgraduate School Master's Thesis
% June 2006
%

dims = length(node_pos(1,:));   % dims = number of rows in node_pos
num_nodes = length(node_pos(:,1));

for count = 1:length(position(1,:));
    distance(:,:,count) = ones(num_nodes,
dims)*diag(position(:,count));
end

for count = 1:length(position(1,:));
    distance(:,:,count) = (distance(:,:,count) - node_pos).^2;
end

b = sqrt(sum(distance,2));
```

B. MATLAB PROGRAMS USED BY PREVIOUS THESES (WITH MODIFICATIONS FOR DATA COMPARISON)

1. Center of Mass Method [4]

```
function fix_pos = complete_program

%This program was used to run simulation one with the center of mass
method
%It also contains all parts to run simulation two and track the glider
with
%real data

global all_nodes range_data;
```

```
% close all %closes all open figure windows
% clear all %clears all variables, functions, etc from memory
% clc       %clears command window


% path(path, '../Read-in excel files');
% % inserted data read-in code from uuv_main()
num_dims = 2;
% % node_file = 'node_gps.txt';
% node_mode = 2;
% node_file = 'sean_nodes.txt';
% % node_file = 'taswex_node.txt';
% all_nodes = node_read(node_file, node_mode);
node_pos = all_nodes;
if num_dims == 3
    heights = [0 -8 0 2 -3 -33]';
    node_pos = [node_pos heights];
end
% % real_node4 = all_nodes(2,:)
% % real_node5 = all_nodes(3,:)
% % real_node6 = all_nodes(4,:)
% % real_node7 = all_nodes(5,:)
% % real_node8 = all_nodes(6,:)
% %
% % all_nodes = all_nodes([2 3 5 6], :);
% num_nodes = length(all_nodes(:,1));

% node_pos=xlsread('program_stationary_nodes'); %node_pos will read
from an
    %excel file to take the positions of the nodes that were used in
the
    %experiment

%-----------------------------------------------------------------------
----
%glider_pos=xlsread('simulation2_inner'); %used for the simulation of
the
                                          %inner track

%glider_pos=xlsread('simulation2_outer'); %used for the simulation of
the
                                          %outer track

%glider_pos=xlsread('program_jul20_glider_ins_pos'); %used to read in
the
    %dead reckoning positioning data from the Glider.
% glider_pos=xlsread('program_jul21_glider_ins_pos');
% glider_pos = dlmread('../../../simulations/pos_grid_truth.txt');
%glider_pos=xlsread('program_jul22_glider_ins_pos');
%-----------------------------------------------------------------------
----

%-----------------------------------------------------------------------
----
```

```
%gps=xlsread('program_jul20_gps_fix'); %Imports excel files with the
gps
% gps=xlsread('program_jul21_gps_fix'); %data for each day
%gps=xlsread('program_jul22_gps_fix');
%----------------------------------------------------------------------
----

% This loop will determine the node to node ranges
(internode_ranges(i,j)),
% which are straight line distances, as well as those ranges translated
to
% the xy plane. This length, internode_ranges_xy (or rij in diagrams),
will
% be less than the straight line distance unless the two nodes are at
the
% same depth.

for(i=1:1:size(node_pos,1));
    for(j=1:1:size(node_pos,1));
        if(i~=j);
            x_diff(i,j)=node_pos(i,1)-node_pos(j,1);
            y_diff(i,j)=node_pos(i,2)-node_pos(j,2);
%             z_diff=node_pos(i,3)-node_pos(j,3);
            internode_ranges(i,j)=sqrt(x_diff(i,j)^2+y_diff(i,j)^2);
%+...
%                 z_diff^2);
%             internode_ranges_xy(i,j)=sqrt((internode_ranges(i,j)^2-
...
%                 z_diff^2));
            internode_ranges_xy(i,j)=internode_ranges(i,j);
        end
    end
end

% G=input...
%     ('Enter the number of times the glider position should be
simulated');

ping_ranges = range_data';   %
dlmread('../../../simulations/rng_grid_a1b0.txt');
ping_ranges_xy = ping_ranges;

G=length(ping_ranges(:,1));

%For simulation 2, G=1:1:size(glider_pos,1).
% for (i=1:1:G); %make a list of positions equal to G for the
simulation
%      glider_pos(i,1)=(.5-rand(1))*2000;
%      glider_pos(i,2)=(.5-rand(1))*2000;
%      glider_pos(i,3)=(rand(1)*100);
% end

% To find the ranges that would be present between the glider and each
node
```

```
% a matrix is created to determine the distances with some random
error, as
% was seen in the experiment. ping_ranges(g,k) looks at the glider
position
% for each (g) and gives the range to each node for that step in the
loop
% (g=1:1:G). An error of up to +/- 10m is used and added to the ranges.
% ping_ranges_xy(1,k) takes the range when transformed down to the xy
plane
% as described in Chapter IV.

for (g=1:1:G);
    clear xgood ygood yoga xoga yogb xogb xogab yogab %This
        %clears all variables that may change in length for different
        %values of g.

    %------------------------------------------------------------------
----
    %ping_ranges=xlsread('program_jul20_ping_ranges'); %These commands
read
        %in files which contain the ping ranges for each day's data
%     ping_ranges=xlsread('program_jul21_ping_ranges');
    %ping_ranges=xlsread('program_jul22_ping_ranges');
    %------------------------------------------------------------------
----

%     for (k=1:1:size(node_pos,1));
%         ping_ranges(g)=sqrt((node_pos(k,1)-glider_pos(g,1))^2+...
%             (node_pos(k,2)-glider_pos(g,2))^2+...
%             (node_pos(k,3)-glider_pos(g,3))^2);
%         %ping_error(g)=0; %No ping error was used for figures 27 and
28
%         ping_error(g)=20*(.5-rand(1)); %Used for a +/- 10m range
error
%         ping_ranges(g)=ping_ranges(g)+ping_error(g);
%         ping_ranges_xy(g,k)=sqrt((ping_ranges(g))^2-(node_pos(k,3)-
...
%             glider_pos(g,3))^2);
%     end

    % Now the ranges between each node as well as the ranges from each
node
    % to the glider have been determined at every step from 1:1:G.
These
    % have also all been translated onto the xy plane. The next step is
to
    % find which triangles form real solutions. While all nodes would
have
    % ranges in this simulation, not all ranges were found at every
time
    % step in the experiment. Solution are only going to be present for
    % range circles that touch, though. The situation described in
Chapter
    % IV, section E, part 1 will not be taken into account since it has
no
    % real affect on the final solution
```

```
        for (i=1:1:size(node_pos,1));
            r1=ping_ranges_xy(g,i);
            for (j=1:1:size(node_pos,1));
                if(i~=j);
                    r2=internode_ranges_xy(i,j); %These give the three
                    r3=ping_ranges_xy(g,j);      %sides of the triangle
                    theta(i,j)=law_of_cosines(r1,r2,r3); %theta will
give
                    %the angle between the two nodes and node i to the
                    %glider(Chapter IV)
                    if(isreal(theta(i,j))==1);
                        theta_good(i,j)=theta(i,j);
                    else
                        theta_good(i,j)=NaN;
                    end
                end
            end
        end

    % Each good pair of nodes (that also have ping ranges) will give an
xy
    % solution. Because of errors and delay times, these positions will
not
    % be the same as for all other pairs' solutions. Factoring all of
them
    % together will lead to a real, approximate solution. Values for
    % theta_good for all solutions at each ping time have now been
    % calculated. The following will calculate phi and then gamma for
each
    % solution.

    count=0;
    for(i=1:1:size(node_pos,1)); %i from 1 to 7
        for(j=i+1:1:size(node_pos,1)); %Only compare the 1st node to
all
            %others and then the second to nodes 3-7, and so on
            if isnan(theta_good(i,j))==0 & (i~=j); %if theta_good is
not
                %NaN, then look at that i,j combination
                count=count+1;
                if (x_diff(i,j)==0); %if both nodes happen to lay
directly
                              %n-s of each other (not necessary for
this
                              %experiment, but may be needed for a
future
                              %experiment, and...
                    if (node_pos(i,2)<node_pos(j,2)); %the y
                        %value of node i is less than the y value of
                        %node j (figure 17)...
                        phi_prime=pi/2; %then phi (the angle between
the x
                            %axis and the line between the two nodes
with
                            %node i as the origin) is pi/2) and...
                        phi(i,j)=phi_prime;
                    else phi_prime=-pi/2; %If the y value of node i is
                        %larger, then phi is -pi/2
                         phi(i,j)=phi_prime;
                    end
                else phi_prime=atan(abs(y_diff(i,j))/abs(x_diff(i,j)));
%If
                    % the x difference is NOT 0 between the two nodes,
                    % and...
                    if ((node_pos(i,1)<node_pos(j,1))&...
                       (node_pos(i,2)<=node_pos(j,2)));
                        phi(i,j)=phi_prime; %In first quadrant
                    elseif((node_pos(i,1)>node_pos(j,1))...
                       &(node_pos(i,2)<=node_pos(j,2)));
                        phi(i,j)=pi-phi_prime; %In second quadrant
                    elseif((node_pos(i,1)>node_pos(j,1))&...
                       (node_pos(i,2)>=node_pos(j,2)));
                        phi(i,j)=pi+phi_prime; %In third quadrant
                    elseif((node_pos(i,1)<node_pos(j,1))&...
                       (node_pos(i,2)>=node_pos(j,2)));
                        phi(i,j)=-phi_prime; %In fourth quadrant
                    end
                end

                gamma_a(i,j)=phi(i,j)+theta_good(i,j); %Each set of
nodes
                    %will give two values of gamma corresponding to the
two
                    %solutions
                gamma_b(i,j)=phi(i,j)-theta_good(i,j);

                % The following part will be used for finding the
solutions
                % using the center of mass method. xoga is the x value
from
                % the center node to the glider for solution a. Since
                % either solution a or b will be good, only one will be
                % kept

                xoga(count)=node_pos(i,1)+ping_ranges_xy(g,i)*...
                    cos(gamma_a(i,j)); %This is the first x value of
the
                        %solution for every i,j combination
                xogb(count)=node_pos(i,1)+ping_ranges_xy(g,i)*...
                    cos(gamma_b(i,j)); %This is the second x value of
the
                        %solution for node i
                yoga(count)=node_pos(i,2)+ping_ranges_xy(g,i)*...
                    sin(gamma_a(i,j));
                yogb(count)=node_pos(i,2)+ping_ranges_xy(g,i)*...
                    sin(gamma_b(i,j));
            end
        end
    end

    % Calculate center of mass for all the points
```

```
    xogab=[xoga xogb]; %Combines all xoga and xogb into one vector
    yogab=[yoga yogb];

    cmx=nanmean(xogab); %nanmean takes the mean of all numbers and
    cmy=nanmean(yogab); %doesn't account for anything that is NaN

    % Now, throw out the point that is farther away from the center of
mass
    % between the two [(xoga,yoga) or (xogb,yogb)]

    count=1; %A counter is started that will only increase by 1 every
        %time a certain condition is met
    for c = 1:count;
        cmadiff(c)= sqrt((cmx - xoga(c))^2 + (cmy - yoga(c))^2);
            %Calculates the distance between the center of mass and
            %solution a
        cmbdiff(c)= sqrt((cmx - xogb(c))^2 + (cmy - yogb(c))^2);
        if cmadiff(c)==NaN    %For an xoga,yoga pair that is not a
number
            %(happens when the range errors shrink the ranges so that
theta
            %is imaginary), make the distance to the center of mass
very
            %large
            cmadiff(c)=10000;
            break
        end
        if cmbdiff(c)==NaN;
            cmbdiff(c)=10000;
            break
        end
        if cmadiff(c) <= cmbdiff(c); %Only use the point that is closer
of
                %the two
            count=count+1;
            xgood(count) = xoga(c);
            ygood(count) = yoga(c);
        else
            count=count+1;
            xgood(count) = xogb(c);
            ygood(count) = yogb(c);
        end
    end

    cmx2=nanmean(xgood); %Take the new center of mass of the good
points
    cmy2=nanmean(ygood);

    good_count=0;
    for c = 1:count;
        cmadiff(c)= sqrt((cmx2 - xoga(c))^2 + (cmy2 - yoga(c))^2);
        cmbdiff(c)= sqrt((cmx2 - xogb(c))^2 + (cmy2 - yogb(c))^2);
         if cmadiff(c)==NaN
            cmadiff(c)=10000;
            continue
        end
        if cmbdiff(c)==NaN;
            cmbdiff(c)=10000;
            continue
        end
        if cmadiff(c) <= cmbdiff(c);
            good_count=good_count + 1;
            xgood(good_count) = xoga(c);
            ygood(good_count) = yoga(c);
        else
            good_count=good_count + 1;
            xgood(good_count) = xogb(c);
            ygood(good_count) = yogb(c);
        end
    end

    % Now, take another center of mass to calculate the final position
    xfinal2(g)=nanmean(xgood);
    yfinal2(g)=nanmean(ygood);

%     sdxfinal=nanstd(xgood);
%     sdyfinal=nanstd(ygood);

    %-----------------------------------------------------------------
----
    %The following part was used to run simulation one with the
weighting
    %method

    %W=zeros(size(xogab,1),1);
    %for(i=1:1:size(xogab,1));
        %for(j=i+1:1:size(xogab,1));
            %factor=100;
            %delta_x(i,j)=xogab(i)-xogab(j); %Find the difference in x
                %position between every solution and every other
solution
            %delta_y(i,j)=yogab(i)-yogab(j);
            %dxy=(delta_x(i,j))^2+(delta_y(i,j))^2;
            %if dxy>=1;         %For two solutions that are far from
                %W(i,j)=(1/dxy) %each other, this is the weight value
            %else               %If they are very close, they get a
weight
                %W(i,j)=factor; %of 100
            %end
        %end
    %end
    %Wsum=sum(nansum(W)); %Take the sum of all of the weights. Since W
                          %will be a 2 dimensional matrix, sum up the
                          %columns first, and then sum those up

    %Each xogab,yogab solution now has a weight. Taking each solutions
    %weight and dividing by the total weight will give a percentage of
    %total weight for that solution. Multiplying that percentage by its
x
    %and y position will contribute to the final solution. The higher
the
```

```
    %weight percentage, the higher the contribution.

    %for i=1:1:length(xogab);
        %for j=1:1:length(xogab);
            %x_fin(i,j)=W(i,j)/Wsum*xogab(i);
            %y_fin(i,j)=W(i,j)/Wsum*yogab(i);
    %end
    %end

    %xfinal2=sum(nansum(x_fin));
    %yfinal2=sum(nansum(y_fin));
    %-----------------------------------------------------------------
----

%     hold on

    %-----------------------------------------------------------------
----
%     plot(glider_pos(g,1),glider_pos(g,2),'b*') %These were all used
to
        % make graphs for real data and for simulation two
%     plot(node_pos(:,1),node_pos(:,2),'ks')
%     plot(xogab,yogab,'ro','MarkerSize',5)
%     plot(xgood,ygood,'ro','MarkerSize',10)


    %-----------------------------------------------------------------
----

%     error(g)=sqrt((xfinal2(g)-glider_pos(g,1))^2+(yfinal2(g)-...
%         glider_pos(g,2))^2); %Used in simulations

end %This closes the primary loop
% plot(xfinal2, yfinal2,'r+:','MarkerSize',10)
% plot(gps(:,1),gps(:,2),'r-.')

fix_pos = [xfinal2; yfinal2]';

% pos_file = ('../../../simulations/sean_pos_out_a095b0.txt');
% out_fid = fopen(pos_file, 'w');
% write_count = fprintf(out_fid, '%g %g\n', fix_pos');
% st = fclose(out_fid);
%
% ave_error=mean(error)
% sd_error=std(error)
% max_error=max(error)
%
% figure(2)
% hist(error,20)
% sorted_error=sort(error); %Sorts all error values
% percent50_error=sorted_error(.5*G)  %These give the percentiles of
error
% percent75_error=sorted_error(.75*G) %for the simulations
% percent90_error=sorted_error(.9*G)
% percent95_error=sorted_error(.95*G)
% percent99_error=sorted_error(.99*G)
```

2. Weighting Method [2]

```
function xy_position = determine_pos

%ENS Matthew Hahn
%6 MAY 2005
%this version of the algorithm takes in an Excel data file with the
ranges from each
%time sample and loops through the rows to garner a solution at each
point in time

global all_nodes range_data;

% close all
% clear all
% clc

% non_mobile_nodes_10_May=xlsread('fixed_node_positions');
% pings_ranges=xlsread('july21_ping_ranges');
%
% Lat=zeros(size(non_mobile_nodes_10_May,1),1);
% Long=zeros(size(non_mobile_nodes_10_May,1),1);
%
% for(k=1:1:size(non_mobile_nodes_10_May,1))
%     Lat(k)=non_mobile_nodes_10_May(k,1) +
(non_mobile_nodes_10_May(k,2)/60);
%     Long(k)=(non_mobile_nodes_10_May(k,3)-
(non_mobile_nodes_10_May(k,4)/60));
% end
%
% %we establish the first node (R10) as the origin and measure all
other node locations
% %relative to it
% node_positions=zeros(size(non_mobile_nodes_10_May,1),2);
% node_positions(1,:)=[0 0];
% for(k=2:1:size(non_mobile_nodes_10_May,1))
%     node_positions(k,2)=(Lat(k)-Lat(1))*111325; %1 degree latitude is
equal to 111km
%     node_positions(k,1)=(Long(k)-
Long(1))*111325*cos(Lat(1)*(pi/180));
% end

node_positions = all_nodes;
pings_ranges = range_data';

n_xy=node_positions;

%it is useful to create a matrix containing all inter-node ranges
%so that they will be available for future calculations
range_set=zeros(size(n_xy,1),size(n_xy,1));
for(i=1:1:size(n_xy,1))
    for(j=1:1:size(n_xy,1))
```

```
        if(i~=j)
            x_diff=n_xy(i,1)-n_xy(j,1);
            y_diff=n_xy(i,2)-n_xy(j,2);
            range_set(i,j)=sqrt(x_diff^2+y_diff^2);
            range_set(j,i)=range_set(i,j);
        end
    end
end

%we now import the Excel file which has the range data for each node at
each
%time sample (rows represent times, column numbers = node numbers)
% ranges=pings_ranges(24:32,:);
ranges=pings_ranges(:,:);

%now we begin the primary loop, which estimates the position of the
mobile node
%for each set of sampled data (assume every 5 min for now, record time
stamps for
%further analysis later)
for(a=1:1:size(ranges,1))

clear pairs R_1 R_2 R_3 theta_a org xax L_o_x delta_x delta_y phi
phi_prime
clear theta gamma_a gamma_b x y W alpha x_diff y_diff x_fin y_fin

% figure(a)
%
% hold on
% %now we need to compute ranges from each fixed node to the mobile
node
% for(i=1:1:size(n_xy,1))
%     %now draw a range circle from the fixed node in consideration
%     %non-available ranges given a value >= 10^10, so we do not plot
%     %ranges circles of magnitude greater than or equal to 10^10
%     if(ranges(a,i)<=(10^10))
%     plot(n_xy(:,1),n_xy(:,2),'ks')
%     draw_circle(ranges(a,i),n_xy(i,1),n_xy(i,2));
%     end
% end



% grid
%for(i=1:1:size(n_xy,1))
 %   text(n_xy(i,1)+15,n_xy(i,2)+15,num2str(i+9));
 %end
%now that we have all possible node pairs (from range__set), we
%determine which ones will yield a real solution for mobile node
position
pairs=zeros(size(range_set,1),size(range_set,2));
for(i=1:1:size(pairs,1))
    for(j=1:1:size(pairs,1))
        if(i~=j)
            R_1=ranges(a,i);
            R_2=ranges(a,j);
            R_3=range_set(i,j);
            theta_a=cosines_law(R_1,R_3,R_2);
            if(imag(theta_a)==0)
                pairs(i,j)=1;
                pairs(j,i)=pairs(i,j);
            end
        end
    end
end

%now we determine the xy coordinates of the two possible solutions for
each pair
%that yields a real solution for theta
c=1;
for(i=1:1:size(pairs,1))
    for(j=i:1:size(pairs,1))
        if(pairs(i,j)==1)
            org=[n_xy(i,1) n_xy(i,2)];
            xax=[n_xy(j,1) n_xy(j,2)];
            L_o_x=range_set(i,j);
            delta_x=xax(1,1)-org(1,1);
            delta_y=xax(1,2)-org(1,2);
            if(delta_x==0) %case of an infinite value of inverse
tangent
                if(org(1,2)>xax(1,2))
                    phi=-pi/2;
                else
                    phi=pi/2;
                end
            else
                phi_prime=atan(abs(delta_y)/abs(delta_x));
                if((org(1,1)>xax(1,1))&(org(1,2)>xax(1,2)))
                    phi=phi_prime+pi; %3rd quadrant
                elseif((org(1,1)>xax(1,1))&(org(1,2)<xax(1,2)))
                    phi=pi-phi_prime; %2nd quadrant
                elseif((org(1,1)<xax(1,1))&(org(1,2)>xax(1,2)))
                    phi=2*pi-phi_prime; %4th quadrant
                elseif((org(1,1)<xax(1,1))&(org(1,2)<xax(1,2)))
                    phi=phi_prime;
                end
            end
            %find abs value of angle of solution offset from new xaxis
            %this is easily determined using the law of cosines,
            %where "opp side" (see function code) is range: xax to
mobile node
            theta=cosines_law(ranges(a,i),L_o_x,ranges(a,j));
            %soln will be endpoint of vector: range from origin,
+origin
            %coordinates, angle equal to gamma (a or b) calculated
already
            gamma_a=phi+theta;
            gamma_b=phi-theta;



            x(c,1)=org(1,1)+ranges(a,i)*cos(gamma_a);
            x(c+1,1)=org(1,1)+ranges(a,i)*cos(gamma_b);
            y(c,1)=org(1,2)+ranges(a,i)*sin(gamma_a);
            y(c+1,1)=org(1,2)+ranges(a,i)*sin(gamma_b);
            c=c+2;
        end
    end
end

%calculate weight values...
W=zeros(size(x,1),1);
for(i=1:1:size(x,1))
    for(j=1:1:size(x,1))
        if(j~=i)
            alpha=2;
            x_diff=x(i)-x(j);
            y_diff=y(i)-y(j);
            W(i)=W(i)+ (x_diff^2+y_diff^2)^-alpha;
        end
    end
end

W_sum=sum(W);

for(i=1:1:size(W,1))
    x_fin(i)=((W(i)/W_sum)*x(i));
    y_fin(i)=((W(i)/W_sum)*y(i));
end
x_position(1,a)=sum(x_fin);
y_position(1,a)=sum(y_fin);

% latitude_soln(1,a)=(x_position(1,a)/111325)+Lat(1);
%
longitude_soln(1,a)=(y_position(1,a)/(111325*cos(Lat(1)*(pi/180))))+Lon
g(1);

% hold on
% %plot(x,y,'r*')
% %plot(x_position,y_position,'g:+')
% plot(x_position(a),y_position(a),'r^')
% hold off


%pause(0.5)
end

xy_position = [x_position; y_position]';

% figure(a+1)
% hold on
% plot(n_xy(:,1),n_xy(:,2),'k*');
% plot(x_position,y_position,'g:+')
% for(i=1:1:size(n_xy,1))
%    text(n_xy(i,1),n_xy(i,2),num2str(i+8));
% end
% grid
% x_position;
% y_position;
% latitude_soln;
% longitude_soln;
```

LIST OF REFERENCES

[1] Kriewaldt, H. A., *Communications Performance of an Undersea Acoustic Wide-Area Network*, M.Sci. thesis, Dept. of Engineering Acoustics, Naval Postgraduate School, Monterey, CA, Mar. 2006.

[2] Hahn, M. J., *Undersea Navigation via a Distributed Acoustic Communications Network*, M.Sci. thesis, Dept. of Engineering Acoustics, Naval Postgraduate School, Monterey, CA, Jun. 2005.

[3] Bowditch, N., *The American Practical Navigator*, 2002 Bicentennial ed., Bethesda, MD: National Imagery and Mapping Agency, 2002.

[4] Ouimet, S. P., *Undersea Navigation of a Glider UUV Using an Acoustic Communications Network*, M.Sci. thesis, Dept. of Engineering Acoustics, Naval Postgraduate School, Monterey, CA, Sep. 2005.

[5] Krause, L. O., "A direct solution to GPS-type navigation equations," *IEEE Transactions on Aerospace and Electronic Systems*., vol, AES-23, no. 2, March 1987, pp. 225-232.

[6] Pickard, G. L., "Sound in the sea," chap. 3.7 in *Descriptive Physical Oceanography*, 5th ed., Woburn, MA: Butterworth-Heinemann, 1990.

[7] Rice, J. A., "A prototype array-element localization sonobuoy," Naval Ocean Systems Center, San Diego, CA, Report No. TR1365, Dec. 1990.

[8] Kreyszig, E., "Linear systems: ill conditioning, norms," sec 19.4 in *Advanced Engineering Mathematics*, 7th ed., New York, NY: John Wiley & Sons, Inc., 1993.

[9] "Length of a degree of latitude and longitude," *Maritime Safety Information* [online, cited June 2006], available from World Wide Web: <http://www.nga.mil/portal/site/maritime/index.jsp>

[10] Rosen, K. H., "Generating permutations and combinations," sec. 4.6 in *Discrete Mathematics and Its Applications*, 1st ed., New York, NY: Random House, 1988.

[11] Mailey, C., “Monterey Bay Glider Operation Notes,” SPAWAR Systems Center, San Diego, CA, 2005.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Joseph Rice
   Naval Postgraduate School
   Monterey, CA

4. Dr. Roberto Cristi
   Naval Postgraduate School
   Monterey, CA

5. Tom Stewart
   The Johns Hopkins University/Advanced Physics Lab
   Laurel, MD

6. Richard Coupland
   Naval Undersea Warfare Center, Newport
   Newport, RI

7. LCDR Steven Whear
   COMSUBLANT
   Norfolk, VA

8. RADM Steve Johnson
   Naval Undersea Warfare Center, Newport
   Newport, RI

9. CAPT Alan Galsgaard
   NAVSEA
   Washington, DC

10. Pete Scala
    The Johns Hopkins University
    Laurel, MD

11. CDR Richard Hale
    NAVSEA
    Washington, DC

12. ENS Sean Ouimet
Naval Aviation Schools Command
Pensacola, FL

13. ENS Matthew Hahn
USS Paul Hamilton, DDG-60
Pearl Harbor, CA

14. Doug Ray
Naval Undersea Warfare Center, Keyport
Keyport, WA

15. Chris Fletcher
Space and Naval Warfare Systems Center, San Diego
San Diego, CA

16. Bill Marn
Space and Naval Warfare Systems Center, San Diego
San Diego, CA

17. Bob Creber
Space and Naval Warfare Systems Center, San Diego
San Diego, CA

18. Doug Grimmett
Space and Naval Warfare Systems Center, San Diego
San Diego, CA

19. CAPT Paul Ims
NAVSEA
Washington, DC

20. Dr. Tom Swean
Office of Naval Research
Arlington, VA

21. Dr. Allan Reed
Technic, Inc.
Cranston, RI

22. Dr. Donald Reed
Ohio Department of Health
Columbus, OH

23. Dr. David Reed
Capital University
Bexley, OH